Task Description for Node.js Interview (MongoDB)

The candidate will be required to set up a basic Node.js application with MongoDB as the database, implementing the following features:

1. Project Setup

- Set up a Node.js application with Express.js.
- Implement middleware for logging requests, error handling, and authentication.
- Use **MongoDB** as the database to store user and todo data, utilizing **Mongoose** for schema definitions and database interactions.

2. User Roles

There will be two roles in the system:

- **Admin**: Can access user data and see the number of todos for each user.
- **User**: Can only create and manage their own todo list.

3. Authentication and Authorization

- Implement **login** and **signup** APIs.
- When a user signs up, their details will be stored in the MongoDB database. Upon login, the system should authenticate users using **JWT** (JSON Web Token) and **middleware** to protect routes.
- For **Admin**, add a pre-defined entry in the database with an admin role to access Admin-only routes.

4. Admin APIs

- The **Admin** role will have access to a `GET /users` endpoint that retrieves a list of all users.
- The response for the user list should include the `numberOfTodo` count for each user (the number of todos associated with that user).

5. User APIs

- **User Registration (Signup)**: Create an API to allow new users to register.
- **User Login**: Implement an API to authenticate users with a JWT token.
- **Todo CRUD**:
    - **Create**: Users can create new todos.
    - **Read**: Users can view their list of todos.
    - **Update**: Users can update the status or details of their todos.
    - **Delete**: Users can delete their todos.
- Each todo should have a `timestamp` field to track when it was created or updated.

6. Database Models (MongoDB with Mongoose)

- **User Model**:

- Fields: `name`, `email`, `password`, `role` (admin or user), and timestamps for creation and last update (using Mongoose's `timestamps: true`).

- **Todo Model**:
  - Fields: `title`, `description`, `status`, `userId` (reference to the User who created the todo), and timestamps for creation and last update.

7. Middleware

- Implement middleware for:
  - **Authentication**: Verifies the JWT token for protected routes.
  - **Role Authorization**: Ensures only Admins can access admin-specific routes, and users can only access their own todos.
  - **Error Handling**: Handles errors and sends appropriate error messages.

8. Expected Deliverables

- Node.js project with MongoDB as the database.
- Well-organized code with proper validation, error handling, and CRUD operations for todos.
- Middleware for authentication and role-based authorization.
- **Timestamps** included in all models for tracking creation and update times.