Full Stack Developer Assessment Task: Task Manager with User Authentication & Profile Management

Overview:

Develop a Task Manager web application that supports full CRUD operations for tasks while integrating user authentication and profile management. This application should allow users to sign up, log in, update their profile, and recover their accounts through "forgot password" and "reset password" functionalities. In addition, use **Redux** to manage state and handle API integration on the front-end.

Project Requirements

1. Front-end

- **Framework:**

  Use **React** or **Next.js** to build the user interface.
- **Features:**
  - **Authentication Pages:**
    - **Registration:** Allow new users to sign up.
    - **Login:** Allow existing users to log in.
    - **Forgot Password:** Provide a form to request a password reset.
    - **Reset Password:** Provide a form to reset the password (accessible via a unique token/link).
  - **User Profile:**
    - Allow authenticated users to view and update their profile information (e.g., name, email).
  - **Task Management:**
    - Display a list of tasks associated with the authenticated user.
    - Provide a form to add new tasks.
    - Allow editing of existing tasks.
    - Enable deletion of tasks.

        - (Optional) Include a detailed view for individual tasks.
- **Design:**
    - Ensure the interface is clean, responsive, and user-friendly.
    - You may use a CSS framework (e.g., Tailwind CSS, Bootstrap) or custom styling.

2. Back-end

- **Framework:**

  Use **NestJS** or **Express.js** to build a RESTful API.
- **Authentication & User Management Endpoints:**
    - **POST /auth/register:** Register a new user.
    - **POST /auth/login:** Authenticate a user and return an authentication token (e.g., JWT).
    - **GET /auth/profile:** Retrieve the authenticated user's profile information.
    - **PUT /auth/profile:** Update the authenticated user's profile.
    - **POST /auth/forgot-password:** Initiate the password recovery process (send reset link/email).
    - **POST /auth/reset-password:** Reset the user’s password using a provided token.
- **Task Management Endpoints:**

  (Ensure that these endpoints are protected and only accessible by authenticated users.)
    - **GET /tasks:** Retrieve a list of tasks for the authenticated user.
    - **GET /tasks/:id:** Retrieve a specific task by its ID.
    - **POST /tasks:** Create a new task.
    - **PUT /tasks/:id:** Update an existing task.
    - **DELETE /tasks/:id:** Delete a task.
- **Error Handling:**

  Implement proper error responses and HTTP status codes (e.g., 400 for bad requests, 401 for unauthorized access, 404 for not found, 500 for server errors).
- **Security Considerations:**
    - Secure password storage (e.g., hashing using bcrypt).
    - Protect endpoints with proper authentication middleware.

- Validate tokens for secure password reset operations.

3. Database

- **Options:**

  Choose **MySQL**, **PostgreSQL**, or **MongoDB** based on your preference.
- **Data Models:**
  - **User Model:**
    - **id:** Unique identifier (auto-generated).
    - **username:** User's display name.
    - **email:** User's email address.
    - **password:** Hashed password.
    - (Optional) Additional fields such as profile picture or bio.
  - **Task Model:**
    - **id:** Unique identifier (auto-generated).
    - **userId:** Reference to the user who owns the task.
    - **title:** A brief title of the task.
    - **description:** Detailed information about the task.
    - **dueDate:** The date by which the task should be completed.
    - **status:** e.g., "pending" or "completed".
- **Validation:**

  Implement both front-end and server-side validation to ensure data integrity.

Optional Enhancements

- **Testing:**

  (Bonus) Write unit and/or integration tests for both the front-end and back-end.
- **Additional Features:**
  - Filtering tasks by status or due date.
  - Pagination for task lists.
  - Enhanced UI/UX features (e.g., animations, real-time updates using WebSockets).

- **Deployment:**
  (Optional) Deploy the application on a hosting platform (e.g., Vercel, Heroku) and include the live URL in your documentation.

Project Deliverables

- **Code Repository:**
  Provide a link to a public Git repository (e.g., GitHub, GitLab) containing your project code.
- **Documentation:**
  Include a README file covering:
- **State Management:** Explain how Redux is used for API integration and managing global state.
    - **Setup Instructions:** How to install and run the project locally.
    - **Overview:** A brief description of your approach and design decisions.
    - **API Documentation:** Detailed explanations of the API endpoints.
    - **Security Measures:** Outline any security practices implemented (e.g., password hashing, token validation).
- **Deployment:**
  (Optional) Share a link to the deployed application.

Evaluation Criteria

- **Functionality:**
  The application meets the core requirements for user authentication, profile management, task CRUD operations, and password recovery flows.
- **Code Quality:**
  Code is clean, well-organized, and follows best practices.

- **Technical Implementation:**
  Effective use of the chosen technologies (React/Next.js and NestJS/Express.js), along with secure handling of authentication and sensitive data.
- **Documentation:**
  Clear and comprehensive setup instructions, API documentation, and explanations of design decisions.
- **User Experience:**
  The UI is intuitive, responsive, and user-friendly.
- **Bonus Enhancements:**
  Additional features, testing, or advanced security measures that exceed the basic requirements.

Submission Guidelines

- **Deadline:**
  Please complete and submit the project **within 16/02/2025.**
- **How to Submit:**
  Share the repository link and any additional deployment URLs reply email.