Project Overview

You will build a full-stack task management application where there are three types of users: **Admin**, **Manager**, and **Regular User**. Initially, only the admin exists in the system. The admin can create new users with the roles of "manager" or "regular user." Managers will have certain privileges, like managing tasks and overseeing users under their management. Regular users can only manage their own tasks.

Key Features to Implement:

User Roles and Management:

1. Initially, the system will only have an **Admin**.
2. The **Admin** can:
    1. Add new **Managers** and **Regular Users**.
    2. Assign specific managers to oversee certain users.
    3. View, edit, or delete any user and their tasks.
3. **Managers** can:
    1. View and manage their own tasks.
    2. View the list of **Regular Users** assigned to them and view or manage their tasks.
4. **Regular Users** can:
    1. Create, edit, and delete their own tasks.
    2. View only their tasks.

Task Management:

5. Users can create, edit, and delete their own tasks.
6. Each task should have a title, description, due date, and status (e.g., pending, in-progress, completed).
7. Users can filter tasks by status and due date.
8. Managers can view and edit tasks created by the regular users they oversee.

Multi-Role Permissions and Access Control:

9. **Admin** can access the entire system: manage all users and tasks.
10. **Manager** can:
    1. Manage only their own tasks.
    2. View and manage tasks of regular users assigned to them.
11. **Regular User** can:

    1. Manage only their own tasks.

12. Ensure each route is protected by the appropriate authorization middleware, checking for roles.

User Authentication and Role Assignment:

13. Use JWT-based authentication.
14. Only registered users (created by the admin) should be able to log in and use the system.
15. Ensure different dashboards are shown based on the user's role:

    1. Admin: Full system view with user and task management.
    2. Manager: View of their tasks and the users they manage.
    3. Regular User: View and manage only their own tasks.

Frontend (React):

16. A dashboard that shows tasks based on the user's role (admin, manager, or user).
17. For Admin:

    1. A user management section where they can create new users (managers and regular users), assign roles, and manage users and their tasks.

18. For Manager:

    1. A dashboard showing their own tasks and the tasks of users they manage.

19. For Regular Users:

    1. A dashboard to manage their tasks.

20. Each user should be able to filter and search their tasks by status and due date.

Backend (Node.js, Express, MongoDB):

21. RESTful API for authentication (login, registration), task management (CRUD for tasks), and user management (admin only).
22. Role-based access control for routes.
23. MongoDB for managing users, roles, and tasks.

Requirements:

- **Frontend**: Built using React. Role-based routing using React Router to show different views for admins, managers, and regular users. Use a state management solution (e.g., Redux or Context API) for global state if needed.
- **Backend**: Use Node.js and Express to build the backend. Implement role-based access control using middleware that protects routes based on the user's role (admin, manager, regular user). Use MongoDB for storing users, roles, and tasks.
- **Authentication**: Implement JWT-based authentication to secure routes and manage sessions.
- **Deployment**: Optionally, deploy the app to platforms like Heroku (for the backend) and Netlify or Vercel (for the frontend).
- **Version Control**: Use Git for version control, and submit the project through a GitHub repository.

Deliverables:

- A GitHub repository link containing the project code.
- Clear instructions in the README on how to run the app locally, including how to set up the backend, frontend, and environment variables.
- Optionally, a deployed version of the app.