**Phase 1: Planning and Initial Setup**

1. **Define Requirements and Design**:
   * Document the app’s core features, such as authentication, project and task management, and the task board.
   * Create a basic wireframe or layout for the user flow, especially for main screens like the dashboard, project view, and task management.
2. **Set Up Version Control**:
   * Create a Git repository and make regular commits to track progress. Use a platform like GitHub for version control and backup.
3. **Environment Setup**:
   * Set up **Node.js** and **Express** for the backend, and initialize the frontend with **Create React App**.
   * Install necessary dependencies like express, mongoose, and cors for the backend, and axios, react-router, and react-beautiful-dnd for the frontend.

**Phase 2: Backend Development**

1. **Database Setup**:
   * Set up MongoDB (local or using MongoDB Atlas) and define Mongoose schemas for **User**, **Project**, and **Task** models.
2. **User Authentication**:
   * Implement user registration and login endpoints.
   * Hash passwords using **bcrypt** and secure authentication using **JWTs** (JSON Web Tokens) to handle user sessions.
3. **API Endpoints for Projects and Tasks**:
   * Build RESTful API endpoints for creating, reading, updating, and deleting **Projects** and **Tasks**.
   * Include error handling and validation to ensure data integrity.
4. **Task Comments and Notifications (Optional)**:
   * Extend the Task schema to support comments and possibly notifications for task deadlines using node-cron.
5. **Testing the Backend**:
   * Test each endpoint using Postman or Insomnia to ensure the API works as expected.

**Phase 3: Frontend Development**

1. **Setup React and Core Pages**:
   * Initialize the React app with pages for **Login**, **Dashboard**, **Projects**, and **Project Details**.
   * Use **React Router** to manage navigation between pages.
2. **Authentication and Authorization**:
   * Set up authentication, storing the JWT in localStorage or cookies.
   * Implement protected routes to restrict access based on user authentication.
3. **Project and Task Management UI**:
   * Create components for the **Project List** and **Project Details** pages.
   * Develop CRUD functionality for tasks using forms or modals for quick edits.
4. **Global State Management with Context API**:
   * Use Context API to manage state for user authentication, projects, and tasks.
   * Ensure the application is responsive and user-friendly by managing reactivity and performance.
5. **Drag-and-Drop for Task Board**:
   * Implement a Kanban board with react-beautiful-dnd for dragging tasks across statuses like "To Do," "In Progress," and "Completed."

**Phase 4: Advanced Features**

1. **Analytics and Calendar View**:
   * Build a dashboard component with metrics like tasks completed, pending tasks, and upcoming deadlines.
   * Add a **Calendar View** for tracking tasks visually with libraries like react-calendar or react-big-calendar.
2. **User Profile (Optional)**:
   * Add a profile page for users to manage their information, with optional role-based permissions.
3. **Styling and Responsive Design**:
   * Style components with **CSS Modules**, **styled-components**, or **Tailwind CSS** for a professional look.
   * Make the app fully responsive on both desktop and mobile views.
4. **Dark Mode (Optional)**:
   * Add a toggle to switch between dark and light themes, storing user preference in localStorage.

**Phase 5: Testing and Deployment**

1. **Testing the Frontend**:
   * Test components and functionality with **Jest** and **React Testing Library** to verify critical workflows.
2. **Final Bug Fixes and Code Review**:
   * Run through the app to fix any bugs, reviewing code for cleanliness and efficiency.
3. **Deployment**:
   * Deploy the **backend** to Heroku or Render and the **frontend** to Netlify or Vercel.
   * Configure CORS settings and environment variables securely.

**Phase 6: Documentation and Portfolio Presentation**

1. **Document the Project**:

Write a clear README.md file with installation instructions, a feature list, and screenshots.

* + Record a demo video to showcase key features and user flow.

1. **Add to Portfolio**:
   * Describe the project on your portfolio site, highlighting technologies used and key features.
   * Include a live link and GitHub repository link for access to the code.