Project Management Tool

Stack: React (frontend) • Node.js + Express (backend) • MongoDB (database)

---

1. Purpose & Summary

A lightweight, scalable project-management web app that supports tasks, projects, teams, comments, file attachments, and basic reporting. Built with React for a responsive SPA UI, Node.js + Express for REST/GraphQL APIs, and MongoDB for flexible document storage. The doc below is a complete plan you can use as a specification, developer-onboarding brief, and exportable PDF.

---

2. Core Features

- **Authentication & Authorization** (JWT + role-based access)
- **Projects**: create, update, archive
- **Tasks**: CRUD, statuses (To Do / In Progress / Done), priorities, due dates
- **Boards**: Kanban view per project
- **Team / Members**: invite, assign roles (Owner, Admin, Member, Viewer)
- **Comments & Activity log** on tasks
- **File attachments** (S3/Cloud storage links)
- **Search & Filters**: by project, assignee, tag, date range
- **Notifications**: in-app + optional email
- **Basic reporting**: task counts, burndown charts

---

3. High-level Architecture

1. **Client (React SPA)**
2. React + React Router
3. State management: React Query / Zustand / Redux Toolkit (choose one)
4. UI library: Tailwind CSS + component primitives (or Material UI)
5. Authentication: store access token (short-lived) + refresh token (httpOnly cookie)
6. **API Server (Node.js + Express)**
7. RESTful endpoints (or GraphQL if you prefer)
8. Auth middleware (JWT verification, role checks)
9. File upload handling (presigned URL flow to S3)
10. Rate limiting, input validation (Joi/Zod)

11. **Database (MongoDB)**

12. Collections: users, projects, tasks, comments, attachments, activity

13. Use indexes for common queries (projectId + status + dueDate)

14. **Aux services**

15. Email service (SendGrid / SES)
16. Object storage (AWS S3 / GCP Cloud Storage)
17. Background jobs (BullMQ + Redis) for notifications, exports
18. Monitoring (Sentry / Prometheus)

---

4. Data Models (example documents)

User

```
{
  "_id": "ObjectId",
  "email": "user@example.com",
  "name": "Full Name",
  "passwordHash": "...",
  "role": "member|admin|owner",
  "createdAt": "ISODate",
  "settings": { }
}
```

Project

```
{
  "_id": "ObjectId",
  "name": "Project Name",
  "description": "...",
  "ownerId": "ObjectId",
  "memberIds": ["ObjectId"],
  "meta": {"archived": false},
  "createdAt": "ISODate"
}
```

Task

```
{
  "_id": "ObjectId",
```

```
  "projectId": "ObjectId",
  "title": "Task title",
  "description": "...",
  "assigneeId": "ObjectId",
  "status": "todo|inprogress|done",
  "priority": "low|medium|high",
  "dueDate": "ISODate",
  "tags": ["frontend","backend"],
  "attachments": ["attachmentId"],
  "createdAt": "ISODate",
  "updatedAt": "ISODate"
}
```

Comment

```
{
  "_id": "ObjectId",
  "taskId": "ObjectId",
  "authorId": "ObjectId",
  "text": "...",
  "createdAt": "ISODate"
}
```

---

5. API Spec (selected endpoints)

All endpoints under `/api/v1`

- `POST /auth/register` — create user
- `POST /auth/login` — returns accessToken (JWT) + set refresh cookie
- `POST /auth/refresh` — rotate tokens
- `GET /projects` — list projects (filterable)
- `POST /projects` — create project
- `GET /projects/:id` — project detail
- `PUT /projects/:id` — update project
- `GET /projects/:id/tasks` — list tasks
- `POST /projects/:id/tasks` — create task
- `PUT /tasks/:id` — update task (status, assignee, etc.)
- `POST /tasks/:id/comments` — leave comment
- `POST /files/presign` — returns S3 presigned URL

Error format: `{ code: string, message: string, details?: any }`

---

6. UI Pages & UX Flow

- **Auth**: login / register / forgot-password
- **Dashboard**: recent projects, assigned tasks, activity
- **Project page**: header with project info, members, filters
- **Board view**: Kanban columns with drag & drop (react-beautiful-dnd)
- **List view**: table with sorting and inline editing
- **Task modal**: details, comments, attachments, change log
- **Team management**: invites, roles
- **Reports**: charts (use Recharts)

---

7. Implementation Notes & Code Snippets

Backend — Express + Mongoose (sample create task)

```
// controllers/tasks.js
const Task = require('../models/Task');

exports.createTask = async (req, res) => {
  const { projectId, title } = req.body;
  const task = await Task.create({ projectId, title, status: 'todo', createdAt:
new Date() });
  // add activity entry, notify assignee (background job)
  res.status(201).json(task);
};
```

Frontend — React + React Query (sample create task)

```
import { useMutation, useQueryClient } from '@tanstack/react-query';

function useCreateTask() {
  const qc = useQueryClient();
  return useMutation(
    (newTask) => fetch('/api/v1/projects/' + newTask.projectId + '/tasks', {
      method: 'POST', headers: {'Content-Type':'application/json'}, body:
JSON.stringify(newTask)
    }).then(r=>r.json()),
    { onSuccess: () => qc.invalidateQueries(['tasks']) }
  );
}
```

---

8. Security & Best Practices

- Store refresh tokens in httpOnly secure cookie. Short-lived access tokens in memory.
- Validate & sanitize all inputs server-side (use Zod or Joi)
- Rate-limit auth endpoints
- Use HTTPS and HSTS in production
- Escape or sanitize rich-text inputs to prevent XSS
- Use CSP headers

---

9. Deployment

- Containerize with Docker; small multi-stage images
- Hosting: AWS (ECS/Fargate or EKS) or Vercel (frontend) + Heroku/Render for backend
- Database: MongoDB Atlas
- Use CI (GitHub Actions) to run tests and deploy

---

10. Testing

- Unit tests: Jest for backend, React Testing Library for frontend
- Integration tests: supertest for API
- End-to-end: Playwright or Cypress

---

11. Observability

- Logging: structured JSON logs, link to request IDs
- Error tracking: Sentry
- Metrics: Prometheus + Grafana or hosted alternative

---

12. Roadmap & MVP scope (4-week sprint example)

Week 1: Auth, User model, basic project CRUD, setup repo, CI **Week 2**: Task model + list views, Kanban UI (drag & drop) **Week 3**: Comments, attachments (presign upload), activity log **Week 4**: Search/filters, basic reporting, polish, E2E tests

---

13. Optional Extensions

- Real-time updates with WebSockets / Socket.IO
- OAuth 2.0 / SSO
- Time tracking & billing

- Advanced reports / integrations (Jira, GitHub)

---

14. Appendix: Helpful Libraries & Tools

- Backend: express, mongoose, bcrypt, jsonwebtoken, bullmq
- Frontend: react, react-router, @tanstack/react-query, react-beautiful-dnd
- Devops: Docker, GitHub Actions, Terraform (infra as code)

End of spec — modify or tell me which sections you want expanded (e.g., full ER diagrams, sequence diagrams, full API spec with request/response examples, or full runnable boilerplate code for client and server).