perplexity

Product Requirements Document (PRD)

Project: Advanced Task Management Backend API (Node.js Only)

1. Overview

Create a scalable, secure, and feature-rich **Task Management REST API** for a project similar to Trello/Asana, exposing endpoints for user and team management, real-time notifications, task workflows, and integrations. The backend is built exclusively in Node.js using modern best practices, following RESTful principles and including real-time capabilities. There will be no front-end; the deliverable is a well-documented API.

2. Objectives

- Provide all core and advanced functionality required for a multi-user, collaborative task management platform.
- Deliver clean, modular, well-tested, and thoroughly documented APIs.
- Support secure authentication, authorization, and scalable data management.
- Support communication via both HTTP REST and WebSockets for real-time updates.

3. Key Features & Scope

3.1 User & Authentication Management

- User registration, login, password reset
- JWT-based authentication
- OAuth2 integration (Google/GitHub)
- Role-Based Access Control (admin, member, guest)
- Email verification endpoints

3.2 Organization & Team Management

- Create/join/leave organizations/teams
- Invite users to teams via email
- Set user roles within teams

3.3 Project, Board & Task Management

- CRUD for projects/boards/lists/tasks
- Task assignment and due dates
- Task attachments (file upload & download endpoints)
- Labels/tags and checklists for tasks
- Task/commenting endpoints

3.4 Real-Time Features

- WebSockets (Socket.IO): task/board updates, assignments, comments, and notifications in real time

3.5 Notifications

- Push in-app notifications (WebSockets) and email notifications (SendGrid or equivalent)
- Event triggers: assignments, mentions, due dates approaching

3.6 Advanced Functionality

- Search and filtering endpoints for tasks/boards
- Background job processing (with Bull/Agenda) for scheduled reminders
- Rate limiting and security middleware
- API versioning
- Environment variable-based config management

3.7 External Integrations

- Email service (SendGrid) for notifications/invites
- SMS gateway (Twilio) for optional SMS notifications

3.8 Documentation & Developer Experience

- Comprehensive Swagger/OpenAPI documentation
- Readme covering setup, usage, and API examples

3.9 Testing

- Unit and integration tests (Jest or Mocha/Chai)
- Coverage reporting

4. Tech Stack

Component	Technology
Backend Runtime	Node.js (LTS)
Server Framework	Express.js
Real-Time	Socket.IO
Database	MongoDB (Mongoose) or PostgreSQL (Sequelize/Prisma)
Caching & Sessions	Redis
Authentication	JWT, Passport.js
Mail/SMS	SendGrid, Twilio
Background Jobs	Bull, Agenda
Documentation	Swagger/OpenAPI
Testing	Jest, Mocha/Chai
Deployment	Docker, Heroku/AWS

5. API Endpoints (Representative Set)

Endpoint	Method	Description	Auth Required
/api/v1/auth/register	POST	Register user	No
/api/v1/auth/login	POST	Login	No
/api/v1/teams	POST	Create new team	Yes
/api/v1/teams/:id/invite	POST	Invite user to team	Yes
/api/v1/projects	POST	Create new project	Yes
/api/v1/boards	POST	Create new board	Yes
/api/v1/tasks	POST	Create new task	Yes
/api/v1/tasks/:id/assign	PATCH	Assign task to user	Yes
/api/v1/tasks/:id/comments	POST	Add comment to task	Yes
/api/v1/notifications	GET	Fetch user notifications	Yes
/api/v1/files/upload	POST	Upload attachment	Yes
/api/v1/search	GET	Search boards/tasks	Yes

(Additional endpoints for update, delete, and advanced filtering to be included.)

6. Non-Functional Requirements

- **Performance:** API latency <200ms for common requests
- **Scalability:** Horizontal scaling supported (stateless server, DB connection pooling)
- **Security:** Input validation, rate limiting, Helmet, CORS, secure data storage
- **Reliability:** Automated tests with >90% coverage, graceful error handling, logging (Winston/Morgan)
- **Documentation:** All endpoints covered in Swagger with example requests/responses

7. Out-of-Scope

- Frontend/UI development
- Mobile app or SDKs
- Analytics dashboards (except basic usage stats endpoint)
- Payment processing

8. Milestones & Delivery

1. **MVP**: User, auth, project, board, task, team management APIs
2. **Real-Time and Advanced Features**: WebSockets, notifications, background jobs
3. **Testing, Caching, and Optimization**
4. **Documentation & Deployment Scripts**

9. Notes

- Code must follow industry-standard patterns and include in-line comments.
- All configuration via environment variables.
- API should be production-ready and easily deployable via Docker.
- Each feature/module should be independently testable and documented.

This PRD ensures your backend project covers all fundamental and advanced Node.js backend subjects. You can iterate from this document, add more modules (like audit tracking or analytics), or scale out features as desired.