Full-Stack Assignment: Task Management Platform

Time Limit: 48 hours
Tech Stack: Node.js OR Python for backend | React for frontend

---

PROBLEM STATEMENT

Build a task management system that allows users to create, organize, and track tasks. The system should support user authentication, task operations with file attachments, collaboration through comments, and provide analytics on task completion.

CORE FEATURES

Backend APIs

Authentication

- Register new user
- Login user
- Get current user profile

Task Operations

- Create new task (with properties: title, description, status, priority, due_date, tags, assigned_to)
- Get all tasks (with filtering, searching, sorting, pagination)
- Get single task by ID
- Update task
- Delete task (soft delete)
- Bulk create tasks

Comments

- Add comment to task
- Get all comments for a task

- Update comment
- Delete comment

File Operations

- Upload files (multiple files per task)
- Get/download file
- Delete file

Analytics

- Get task overview statistics (counts by status, priority)
- Get user performance metrics
- Get task trends over time
- Export tasks data

Technical Requirements

- RESTful API design with proper HTTP methods and status codes
- Input validation on all endpoints
- Comprehensive error handling
- API documentation (Swagger/Postman)
- Database schema with proper relationships and indexes
- Authentication and authorization on protected routes
- File type and size validation
- Security: CORS, rate limiting, input sanitization

Frontend

Pages & Features

- Authentication (login/register)
- Dashboard with overview statistics
- Task list with filtering and search
- Task detail view
- Task creation and editing
- User profile
- Analytics/reports page

UI/UX Requirements

- Responsive design
- Form validation
- File upload with drag-and-drop

- Loading and error states
- Empty states
- Confirmation dialogs

Data Visualization

- Charts for task statistics
- Trend visualizations
- Performance metrics

Technical Requirements

- React with hooks
- State management
- Client-side routing
- TypeScript
- Custom styling (no CSS frameworks)
- Performance optimization

BONUS FEATURES

- Real-time updates (WebSockets)
- Email notifications
- Background job processing
- Caching layer
- Markdown support in comments
- Dark mode
- Testing suite
- Docker setup

DELIVERABLES

1. Source code pushed to GitHub
2. Comprehensive README with:
    - Setup instructions
    - Environment variables
    - How to run the application
    - Architecture decisions
    - Assumptions made
3. API documentation

4. Demo video (5-10 minutes) showing:
    - Application walkthrough
    - Code architecture explanation
    - Technical challenges solved
5. Test user credentials

SUBMISSION FORMAT

Repository URL(s): [GitHub links]
API Documentation: [Link]
Demo Video: [Link]

NOTES

- Use any libraries/tools within the specified tech stack
- Document assumptions if requirements are unclear
- Focus on working software with good documentation
- Manage time wisely - core features before bonuses

Good luck! □