# 🚀 Project: Personal Notes & Bookmark Manager

- Tech Stack
  - Backend: Node.js + Express + MongoDB/PostgreSQL
  - Frontend: Next.js / JavaScript (React) + Tailwind CSS

- Objective
  - Build a Personal Notes and Bookmark Manager app where users can:
    Save and search notes with tags
    Save bookmarks with URLs, titles, and descriptions
    Filter and search items

    (Bonus): Auto-fetch metadata (title) from bookmark URLs.

- Requirements
  - Backend (Node.js + Express)
    Build a REST API with these routes:
    Notes API
    POST /api/notes
    GET /api/notes (optional query: ?q=searchTerm&tags=tag1,tag2)
    GET /api/notes/:id
    PUT /api/notes/:id
    DELETE /api/notes/:id

    Bookmarks API
    POST /api/bookmarks
    GET /api/bookmarks (optional query:
    ?q=searchTerm&tags=tag1,tag2)
    GET /api/bookmarks/:id
    PUT /api/bookmarks/:id
    DELETE /api/bookmarks/:id

    General
    Required field validation
    URL validation for bookmarks
    Proper HTTP status codes and error messages
    (Bonus): Auto-fetch title if user leaves the bookmark title empty

  - Frontend (Next.js + Tailwind CSS)
  - Build a responsive and clean UI.
  - Pages
    /notes – list/search notes
    /bookmarks – list/search bookmarks

- Features
  - Create, update, delete notes and bookmarks
  - Filter by tags or search text
  - Mark favorites (optional)

    - Responsive UI using Tailwind

- Bonus Features
    - User authentication (JWT)
    - Show data only for the logged-in user
    - Mark notes/bookmarks as favorites

- Deliverables
    - Backend
        Node.js project with /api-prefixed routes
        Clear folder structure
        Setup instructions in README.md
    - Frontend
        Next.js + Tailwind project
        Pages /notes and /bookmarks
        Responsive design
        Setup instructions in README.md
- README
    - Project setup steps (backend + frontend)
    - Brief API documentation

    (Optional) Postman collection or sample cURL requests

    - Skills This Tests
    - REST API design
    - Data validation and error handling
    - React (Next.js) routing and state
    - Tailwind CSS for UI
    - Clean code and structure
    - Real-world data modeling
- Evaluation Criteria
    - Working CRUD functionality
    - Code structure and modularity
    - Error handling and validation
    - UI/UX quality (responsive and clean)

    Clarity in README and setup instructions

- Submit Your Assignment
    - Deadline: Within 24 hours of receiving this challenge
    - Submit via this form: Link