Project Overview: Web-Based To-Do List Application

Objective

Develop a full-stack website that allows users to create, organize, and manage categorized to-do lists. The project will focus on building a functional web app using MongoDB for persistent data storage, developing both frontend and backend components, and gaining experience with APIs, web frameworks, and deployment.

Learning Goals

• Gain hands-on experience with MongoDB for database design and integration.

• Strengthen web programming and styling skills (HTML, CSS, JavaScript, frontend frameworks).

• Learn how to integrate and use APIs (with a stretch goal of implementing Google login authentication).

Core Features

- Multiple To-Do Lists
  - Users can create and manage separate lists by category (e.g., Cooking, School, Household).
- Task Management
  - Add new tasks with a title and description.
  - Save tasks to the MongoDB database so that data persists after reload.
  - Mark tasks as completed (highlight in green) (if you could make the task have a strikethrough effect that would be cool, but not needed)
  - Delete or edit existing tasks
  - Click on a task to expand and view more details (integrated with edit tasks)
- User Interface
  - Main page containing multiple to-do lists
  - Dedicated to-do list pages for individual categories

Tech Stack

- Frontend: Basic framework (to be chosen after research; Node.js-compatible frontend such as React or Vue recommended).
- Backend: Node.js server connected to a MongoDB database.
- Database: MongoDB.
- Optional (Stretch Goal): Implement Google authentication via Google API.

Progress Tracking

- Submit screenshots for visual milestones (UI updates).
- Submit screen recordings for feature milestones (functional updates).
- Notify Ethan when a new feature is implemented.
- Keep code modular; reuse universal elements.
- Break large scripts into smaller, manageable files.

Timeline

• Project Duration: ~3 weeks

Development Plan

Phase	Description	Deliverable
1. Prototype	Hi-FI prototype of the website layout and page flow.	Screenshots
2. Basic Frontend	Build a functional website with core structure, no styling.	Screenshots
3. Styled Frontend	Add CSS or framework-based styling.	Screenshots + Github
4. Backend Integration	Connect frontend to MongoDB; ensure data persistence.	Recordings + Github
5. Features	- Add edit/delete - Google login	Github + Recordings
6. Another Feature	Email Reminders	Github
7. Deployment	Deploy finished project	Live URL

Project Notes and Tips

Google login → create database of people who have log in (preferably)

- If you want simplicity, do preadmin (predetermined list of users)

- Filtering system → searching (easier)
    - Home page → search bar (auto update)
    - search a list or specific task
        - ➢ If you search a specific task, it brings up the list and task
- Add a notification feature (email reminders)
    - Research email api
    - Connect api to website
    - Add 2 things to “add task pop up”
        - ➢ Add date (and probably time) to task
            - Need to add a script that updates all previous tasks to have a date tag
        - ➢ Send a reminder (to email that the account is made with)
            - Toggle: By default should be off
            - Email Content: List Name, Task Added, All Previous Tasks (that are NOT completed)
- Email api (decided on Resend)
    - Use one that is more versatile and has a lot of freedom
    - Format of the email is up to me (can be simple text reminder)
    - Might have to make an email for listhub (sender, noreply email)
    - Might have to authenticate the sender email