Fluidify – Task Management React Application

Project Proposal:

1. Project Overview:

Fluidify is a modern, intuitive task management application designed to help individuals enhance productivity, stay organized, and manage their time efficiently. Inspired by the concept of seamless flow, Fluidify provides users with a smooth and clutter-free experience that allows them to "Flow Through Your Day."

Built with a user-first mindset, **Fluidify** aims to eliminate the chaos of task overload by offering a visually appealing and functionally powerful tool. From daily to-dos to project milestones, Fluidify adapts to users' needs while promoting balance, clarity, and progress.

2. Project Objectives:

- Simplify task organization through a clean, minimal interface and intuitive workflows.
- Promote productivity and time management with smart scheduling and priority tagging.
- Deliver a smooth user experience by combining elegant UI/UX design with robust performance.
- ensure accessibility across platforms via responsive web design.

3.Project Scope:

The initial development phase of **Fluidify** will include the following core features:

- Task Management:
  - Create, edit, delete tasks
  - Set due dates, priorities, and reminders
  - Tag tasks with categories or labels
- User Dashboard:
  - Overview of today's tasks and progress
  - Quick access to priority items and upcoming deadlines
- Notifications & Reminders:
  - Smart reminders based on task urgency.
  - Push/email notifications (into consideration depends on how tight the deadline will be)

4.Tech Stack:

- **Frontend**: React, Tailwindcss and javascript
- **Backend**: Node.js and Express.js
- **Database**: MongoDB "on Atlas"

Project Plan:

Timeline Overview:

Phase	Duration	Timeline
Project Setup & Planning	5 days	Mar 21 – Mar 25
UI/UX Design (Wireframes + Prototypes)	1 week	Mar 26 – Apr 1
Frontend Development (React)	3 weeks	Apr 2 – Apr 22
Backend Development (API & DB)	3 weeks	Apr 2 – Apr 22 (parallel)
Task Management & Dashboard Integration	1 week	Apr 23 – Apr 30
Testing & Bug Fixing	1.5 weeks	Apr 30 – May 9

Milestones:

Milestone	Date
Project Kickoff	Mar 21, 2025
Design Completion	Apr 1, 2025
Frontend Core Functionalities Ready	Apr 22, 2025
Backend API and Database Connected	Apr 22, 2025
Task Management Features Integrated	Apr 30, 2025

Deliverables:

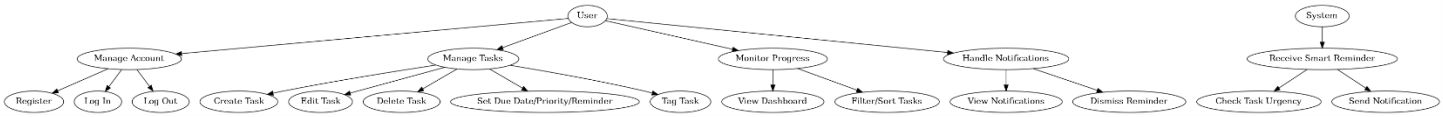
- Wireframes & UI Prototype
- Responsive Frontend (React + TailwindCSS)
- Backend API (Node.js + Express)
- MongoDB Atlas Database Structure
- Task Management System
- User Dashboard Interface
- Notification System (initial version)

Task Assignment & Roles:

As this project will be executed by a single individual, a formal task assignment breakdown by team roles is not applicable. All responsibilities—including project planning, UI/UX design, frontend and backend development, database management, and deployment—will be managed and carried out by the developer independently.

Use Case Diagram & Descriptions:

Use Case Diagram:



Actor:

- User (Primary actor: interacts with all features).

Use Cases (Core Functionality):

- **Manage Account**
  - Register
  - Log In
  - Log Out
- **Manage Tasks**
  - Create Task
  - Edit Task
  - Delete Task
  - Set Due Date/Priority/Reminder
  - Tag Task with Category
- **Monitor Progress**
  - View Dashboard (Today's Tasks, Progress Overview, Upcoming Deadlines)
  - Filter/Sort Tasks (by Date, Priority, or Tag)
- **Handle Notifications**
  - View Notifications
  - Dismiss Reminder

Key Relationships

- The **User** interacts with all task management and dashboard features.
- Notifications are triggered automatically by the **System** based on task data.

Functional & Non-Functional Requirements:

Functional (Capabilities):

- Users can create/edit/delete tasks with due dates, priorities, and tags.
- Dashboard displays today's tasks, progress metrics, and upcoming deadlines.
- Notifications for reminders (email/push).
- Responsive UI works on all devices (TailwindCSS).
- Secure authentication (JWT, password hashing).

Non-Functional (Constraints):

- Performance: Dashboard loads in ≤2s, API latency ≤1s.
- Security: HTTPS, MongoDB encryption.
- Scalability: Backend handles 1,000 concurrent users.

Software Architecture

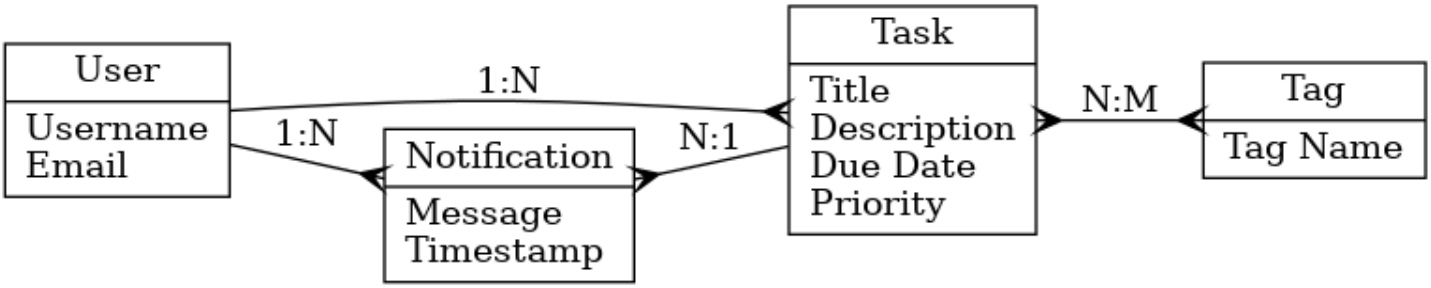
Style: Layered (MVC) for separation of concerns.

- **Frontend**: React (UI components, state management).
- **Backend**: Node.js/Express.js (REST APIs).
- **Database**: MongoDB Atlas (NoSQL for flexible task data).

Interactions:

- React ↔ Express API ↔ MongoDB.

Database Design & Data Modeling



Entities:

- **Users**
- **Tasks**
- **Tags**
- **Notifications**

Relations:

- **User - > Tasks 1:N**
- **User -> Notifications 1:N**
- **Task - > User N:1**
- **Task -> Tags N:M**
- **Tag - > Tasks M:N**
- **Notifications - > Tasks N:1**
- **Notifications - > Users N:1**

Data Flow & System Behavior

DFD (Context-Level):

- User → React App → Express API → MongoDB Atlas.

Sequence Diagram (Create Task):

- User submits task → React → Express → MongoDB → Confirmation.

Activity Diagram:

- Flow: Log in → Dashboard → Create Task → Set Reminder → Log out.

State Diagram (Task):

- States: Pending, Completed, Overdue.

Class Diagram:

- User: userId, email, password.
- Task: taskId, title, dueDate, priority.

UI/UX Design & Prototyping:

The UI/UX design phase is currently under thoughtful development, with ongoing exploration of ideas to craft a clean, intuitive, and user-friendly interface. The visual direction is being refined to ensure a seamless experience that aligns with the core values of clarity, productivity, and ease of use. The goal is to deliver a minimalistic yet engaging design that enhances task management efficiency while maintaining aesthetic appeal.