Project Name – NoteNest

Team Name –

Team Members –

1. Aditya Pise (anpise@iu.edu)
2. Mohammed Pathariya (mjpathar@iu.edu)
3. Nileet Savale (nsavale@iu.edu)

Motivation

Everyone loves taking notes, whether it is a quick idea that strikes during work, a grocery list, a project reminder, or a learning summary. We use different tools for each such as sticky notes, phone reminders, Google Keep, Notion, or random documents, and often end up writing things down somewhere only to forget where we put them later.

Over time, these scattered notes, reminders, and to-dos become a mess to handle. Many of us also dislike adding too many details or structuring every note manually, which makes note management even harder.

These pain points inspired us to create NoteNest, a smart notes management system that understands and organizes notes automatically. The goal is simple: you just write naturally, and NoteNest will categorize it for you, remind you when needed, and present everything neatly in a dashboard where you can track, revisit, or archive notes with ease.

Proposed Solution

NoteNest will be a simple web application with a main dashboard. It will have a chat box where users can type their notes in normal language. Each note will be automatically placed into a

predefined category using an LLM. Users can also view all categories, edit them, or create their own with a short description.

On the dashboard, users can see basic stats like how many notes are in each category and recent activity. Every note will show its date and options to delete or update it. These actions can also be done using simple natural language, such as “show my notes from today” or “delete my meeting note.”

This makes it easy to add, organize, and manage notes without any extra effort.

Data Description

Data Source

The data for NoteNest will be created by our team. Each team member will add their own sample notes, categories, and reminders to build and test the system. This approach helps capture different note-taking styles and user behaviors for more realistic results.

Data Storage

We will use a NoSQL database such as MongoDB for storing data. It allows flexible storage of notes in various formats without the need to create or manage complex tables. Each note can include text, category, date, tags, and optional reminders in one document, making it easier to manage and update dynamically.

Team Roles & Skills

1) Aditya Pise

- Skills –
    - Python, TypeScript, FastAPI, React
    - MongoDB, DynamoDB
    - LLMs, OpenAI, Claude API
    - Backend APIs, Frontend UI
- Team Role –
    - Managing the project milestones and deliverables
    - Assigning tasks to team members
    - Creating architecture of application
    - Backend APIs and integration with LLMs

2) Mohammed Pathariya

- Skills
  - Python, SQL
  - Machine Learning (Scikit-learn, Pandas, NumPy)
  - Data Analysis & Visualization
  - Database Design (Relational/SQL and NoSQL)
  - Backend APIs (FastAPI)
- Team Role
  - Leading the product vision and feature requirements.
  - Co-designing the database schema
  - Developing and testing core backend API endpoints.
  - Implementing the data analytics and visualization components (e.g., writing the aggregation queries for the dashboard).
  - Assisting with final deployment and project documentation.

3) Nileet Savale:

- Skills
  - Python and FastAPI
  - MongoDB and data organization
  - JavaScript, HTML, CSS
  - Data visualization (Chart.js / D3.js)
- Team Role
  - Building and connecting the MongoDB database for storing notes and tags
  - Developing backend CRUD operations using FastAPI
  - Creating the frontend dashboard for adding, editing, and searching notes
  - Designing data visualizations for note activity and tag usage
  - Assisting with testing and final deployment setup

Technology Stack

Frontend:

- React.js
- HTML, CSS, JavaScript

Backend:

- Python FastAPI
- RESTful APIs

Database:

- MongoDB (NoSQL)

AI Integration:

- LLM API (e.g., OpenAI / Claude)

Milestones (6 Weeks)

Week 1: Design database schema and system architecture
Week 2: Build Flask backend and connect MongoDB
Week 3: Set up React frontend and basic dashboard
Week 4: Add chat box, auto-categorization, and category management
Week 5: Add dashboard stats and natural language actions
Week 6: Prepare demo, presentation, and final report

References & Resources

- MongoDB Documentation
- MongoDB in Python: Building and Querying Databases – DataCamp Course
- **LangChain Documentation** – for integrating LLMs and building smart workflows
- **FastAPI Documentation** – for developing high-performance backend APIs
- **OpenAI API Documentation** – for natural language processing and note categorization
- **React.js Official Docs** – for building modern and responsive frontend interfaces