HACKATHON 2: RAG & Vector Databases (Days 4–6)

Objective: Grounding AI in custom data using MongoDB Atlas Vector Search.

1. MindVault: Semantic Personal Notes (Productivity)

Description: Traditional note apps fail when you forget the exact words you used. **MindVault** is a second-brain application that understands the *intent* and *context* of your thoughts. It uses mathematical representations (embeddings) to connect related ideas, even if they share zero overlapping keywords.

- **Detailed User Workflow:**
  - **Capture:** The user types a note in a React-based editor (e.g., "The sunset at the beach yesterday was calming") and hits save.
  - **Vectorization (Background):** The Node.js backend sends this text to an embedding model (like OpenAI's `text-embedding-3-small`). The model returns a vector (a list of numbers like `[0.12, -0.04, ...]`).
  - **Storage:** The note and its vector are stored in a **MongoDB Atlas** collection with a Vector Index.
  - **The "Vibe" Search:** Months later, the user searches for "peaceful evening memories."
  - **Semantic Retrieval:** The system converts the search query into a vector and uses the `$vectorSearch` operator to find the note about the sunset because "peaceful" and "calming" are mathematically close in vector space.
- **Core Features:**
  - **Vector Search Integration:** Implementing the bridge between a MERN backend and MongoDB Atlas Vector Search.
  - **Smart Query Interface:** A search bar that handles natural language instead of strict tags or keywords.
- **Optional Features:**
  - **"Related Notes" Sidebar:** As the user types a new note, a side panel automatically refreshes to show similar past notes, helping the user connect old ideas to new ones in real-time.
  - **Auto-Categorization:** The AI suggests tags (e.g., #Travel, #Health) based on the content of the note.

2. RuleBook AI: Corporate Q&A (Enterprise)

Description: Employees rarely read 100-page HR or Travel policy manuals. **RuleBook AI** turns these static PDFs into an interactive consultant. It solves the "Hallucination Problem" by forcing the AI to only answer using the provided text, making it safe for corporate use.

- **Detailed User Workflow:**

    - **Ingestion:** An Admin uploads a company PDF (e.g., "Employee_Handbook_2025.pdf").
    - **The "Chunking" Pipeline:** The Node.js server breaks the PDF into smaller, overlapping segments (e.g., 500 words each) so the AI doesn't lose context.
    - **Indexing:** Each chunk is embedded and stored in MongoDB with metadata (page number and section header).
    - **Questioning:** An employee asks, "Can I claim a high-speed internet bill while working from home?"
    - **Grounded Response:** The system retrieves the 3 most relevant chunks from the handbook and sends them to the LLM with a prompt: *"Answer only using the context below. If not mentioned, say you don't know."*
    - **Citations:** The UI displays the answer along with a "Source" badge (e.g., "See Page 14, Section: Remote Work").
- **Core Features:**
    - **PDF Processing:** Using libraries like `pdf-parse` or `langchain` to handle document structure.
    - **RAG (Retrieval-Augmented Generation):** The core logic of combining database retrieval with LLM generation.
    - **Source Citation:** Passing metadata through the pipeline to prove where the answer came from.
- **Optional Features:**
    - **Admin Analytics Dashboard:** A React dashboard showing a "Word Cloud" or list of common employee questions, helping HR identify which policies are confusing.
    - **Multi-Doc Library:** Allow users to toggle between different manuals (e.g., "Insurance Policy" vs. "Code of Conduct").

3. Clinical SOP Guide (Healthcare)

Description: In a hospital setting, following the wrong procedure can be fatal. This tool acts as a "Digital Quality Controller" for medical staff. Unlike a general chatbot, this system is **strictly grounded**, meaning it is programmed to say "I don't know" rather than guess, ensuring that nurses and doctors get 100% factual data from their internal Standard Operating Procedures (SOPs).

- **Detailed User Workflow:**
    - **Selection:** A nurse on a tablet selects the "Sterilization & Hygiene" manual from a list of hospital departments.
    - **Query:** The nurse types a specific question: "What is the post-op cleaning protocol for an orthopedic suite?"
    - **Vector Retrieval:** The system searches the MongoDB Vector database for the most relevant chunks of the "Infection Control" PDF.
    - **Verification (The Guardrail):** The backend uses a "System Prompt" that tells the LLM: *"You are a medical safety assistant. Use ONLY the provided context. If the*

    answer is not in the text, respond: 'This information is not in the official SOP. Please consult your supervisor.'"
  - **Direct Answer:** The AI provides a bulleted list of steps extracted directly from the manual, accompanied by the document's official "Last Updated" date.
- **Core Features:**
  - **Strict Grounding Logic:** Implementing "Hallucination Protection" by engineering prompts that forbid the AI from using its general training data.
  - **Source Verification:** Every answer must link to a specific section of the PDF (e.g., "SOP #402, Section B").
- **Optional Features:**
  - **"Change Log" Comparison:** A feature where the AI compares an old SOP (2023) with a new one (2024) and highlights exactly what changed (e.g., "The required alcohol concentration was increased from 60% to 70%").
  - **Emergency Mode:** A high-contrast UI for rapid-fire questions during critical situations.

4. Fin-Doc Semantic Search (FinTech)

Description: Financial analysts spend hours digging through 200-page earnings reports to find specific insights. This project builds a "Financial Intelligence Engine" that allows users to perform **Hybrid Search**—combining the power of AI meaning (Semantic) with hard filters like dates, tickers, and document types (Metadata).

- **Detailed User Workflow:**
  - **Ingestion:** The user (an analyst) uploads several "10-K" or "Quarterly Earnings" PDFs for multiple companies (e.g., Apple, Tesla, NVIDIA).
  - **Metadata Tagging:** As files are uploaded, the MERN backend stores them with tags like `year: 2024`, `company: Tesla`, and `documentType: Earnings`.
  - **Complex Search:** The user searches for "AI infrastructure investment" but applies a filter for "Only 2024" and "NVIDIA."
  - **Hybrid Retrieval:** The system performs a MongoDB `$vectorSearch` but *pre-filters* the data using the metadata tags, ensuring the AI only looks at the correct documents.
  - **Insight Summary:** The AI doesn't just find the text; it summarizes the "Growth Outlook" based on all the filtered snippets it found.
- **Core Features:**
  - **Metadata Filtering:** Learning how to combine traditional MongoDB queries (Year/Company) with Vector Search.
  - **Ranked Results:** Displaying search results with a "Similarity Score" to show how closely the document matches the query.
- **Optional Features:**
  - **Sentiment Trend Analysis:** The AI analyzes the retrieved chunks and determines if the "tone" of the executives is becoming more "Bullish" (positive) or "Bearish" (negative) compared to previous reports.

    - **Comparison Dashboard:** A React view that puts the AI-summarized "Risk Factors" of two different companies side-by-side.

5. SecureCode Knowledge Base (Cybersecurity)

Description: In high-security environments, developers cannot use public AI for sensitive code. **SecureCode** is a localized, private knowledge base that stores an organization's "Gold Standard" security practices. It ensures that every developer—from junior to senior—implements security-first code based on internal compliance, not generic (and potentially vulnerable) internet snippets.

- **Detailed User Workflow:**
    - **Ingestion:** A Security Lead uploads "Security Standards" markdown files or PDFs (e.g., "OWASP Top 10 Prevention in Node.js").
    - **The Query:** A developer asks, "What is our company's approved method for encrypting JWTs?"
    - **Semantic Retrieval:** The system searches the MongoDB Vector database for the specific encryption standards.
    - **Code-Centric Response:** The AI returns a "Code + Explanation" block. It provides the exact code snippet required and explains the security headers used.
    - **Validation:** The AI cross-references the query against "Forbidden Practices" (e.g., if the user asks for MD5, the AI explicitly warns: "MD5 is deprecated; use Argon2 per company policy").
- **Core Features:**
    - **Syntax-Aware RAG:** Ensuring the system retrieves and renders code blocks correctly in the React UI using libraries like `react-syntax-highlighter`.
    - **Secure vs. Unsecure Logic:** The system provides two code blocks: "Don't do this" (vulnerable) vs. "Do this" (secure).
- **Optional Features:**
    - **Mock GitHub Integration:** A "Scan my Code" button where a user pastes a link to a mock repo; the AI retrieves the file via GitHub API and checks it against the SecureCode Knowledge Base for vulnerabilities.
    - **Language Toggle:** Users can filter results to only show code in Python, JavaScript, or Go.

6. AI Recipe "Vibe" Finder (Lifestyle)

Description: Traditional recipe apps are limited to ingredients. **Vibe Finder** uses "Conceptual Search." It understands the emotional and situational context of food. Instead of searching for "Tomato Soup," a user can search for "Something my grandmother would make to cure a cold," and the AI connects the *vibe* of the query to the *flavor profiles* in the database.

- **Detailed User Workflow:**

    - **Recipe Scraping:** The user uploads a PDF of a scanned family recipe or pastes a URL from a food blog.
    - **Flavor Profiling:** During the embedding process, the LLM generates "Hidden Tags" for the recipe (e.g., "Savory," "Umami," "Comfort Food," "Rainy Day").
    - **The Search:** The user types a "mood" search: "I'm stressed and want something crunchy but healthy."
    - **Vector Match:** The system calculates the distance between "Stressed/Crunchy" and the "Flavor Profiles" in MongoDB Atlas.
    - **Interactive Recipe Card:** The UI shows the recipe, a "Vibe Match" score, and a generated summary of why this recipe fits their mood.
- **Core Features:**
    - **Multi-Modal Metadata:** Storing ingredients, instructions, and "Vibe Tags" in a single MongoDB document.
    - **Semantic Query UI:** A React interface that encourages descriptive, mood-based searching.
- **Optional Features:**
    - **"Substitute Finder" (Agentic RAG):** If the user is missing an ingredient, they click "Substitute." The AI searches other recipes in the database to see what was used in similar contexts (e.g., "You're out of Eggs; your 'Vegan Brownies' note suggests using Flax-meal").
    - **Meal Planner:** AI suggests 3 recipes that "share a vibe" for a 3-course dinner.

7. Real-Estate Contract Scout (PropTech)

Description: Reviewing multiple rental or purchase agreements is a manual, error-prone task. **Contract Scout** is a "Comparison Engine" that allows a user to "Parallel Search" across a library of legal documents. It is designed to extract specific data points into a structured format, making it easy to spot the best deal or the hidden risk.

- **Detailed User Workflow:**
    - **Batch Upload:** A user (or real estate agent) uploads 5–10 PDF rental agreements for different properties.
    - **Parallel Indexing:** The system creates a "Collection" in MongoDB where each chunk is tagged with a `propertyID`.
    - **Cross-Document Query:** The user asks: "Compare the pet policies and security deposits for all properties."
    - **Data Extraction:** The system retrieves the relevant sections from *all* 10 documents.
    - **Comparative View:** The React frontend renders a **Comparison Table**. The LLM populates the table: (Property A: $500 Deposit, Pets Allowed | Property B: $1000 Deposit, No Pets).
- **Core Features:**
    - **Multi-Document RAG:** Learning how to query across different `sourceID` tags in a single vector search.

    - **Structured Table Generation:** Forcing the LLM to output JSON that can be mapped directly into an HTML `<table>`.
- **Optional Features:**
    - **Red-Flag Scout:** Automatically highlights "Illegal Clauses" (e.g., a landlord asking for 3 months' rent as a deposit when the local law only allows 1).
    - **Lease Expiry Tracker:** AI extracts the "Termination Date" and "Notice Period" to create a calendar view in the dashboard.

📘 Hackathon 2 Study Prompt: "Teach me: 1) What are embeddings? 2) How does MongoDB Atlas Vector Search work? 3) How do I chunk long PDFs for RAG?"
