Capstone Project Proposal – Spring 2026

MindMapper: Visual Multi-Context Interfaces for LLM Interaction

Dipan Bag
bag00003@umn.edu

Abstract

Current large language model (LLM) tools—like ChatGPT, Claude, and Gemini—operate within linear, scroll-based interfaces. This structure becomes unwieldy in complex reasoning or exploratory sessions, where users must scroll excessively or start fresh chats to explore new branches. MindMapper proposes a canvas-based, node-driven interface for LLM interaction. Users create query nodes on a visual canvas, and responses form directional links that can be branched further. Each path holds a distinct context, with local memory maintained via Retrieval-Augmented Generation (RAG) using FAISS embeddings. The system blends cognitive research on diagrammatic reasoning with UX strategies inspired by Miro, Obsidian Canvas, and NotebookLM. This project will deliver a working prototype and evaluate usability, clarity, and memory control benefits in a user study.

1 Introduction

Motivation

The inspiration for MindMapper arose from a common frustration many users experience with AI chat tools such as ChatGPT and Claude. While these systems are powerful for ideation and research, their linear chat format makes it difficult to revisit or branch off earlier discussions. When users want to explore a follow-up idea buried dozens of messages above, they often have to scroll back and lose reading context or restart a new chat entirely. Over time, this constraint forces nonlinear thinking into a one-lane thread — limiting creative exploration.

I imagined: what if thoughts and prompts could be placed like post-its on a board, expanded, and forked into new directions without breaking flow? What if each follow-up could have its own thread, with its own memory and context? That reflection led to the conception of MindMapper.

Background

Chat interfaces dominate LLM UX, but they offer poor support for non-linear reasoning. Each conversation is a scroll thread with a shared, often fragile context. As discussions grow longer, earlier details are forgotten, responses drift off-topic, and new directions must be awkwardly shoehorned into the same thread or started from scratch [1].

Recent updates, like OpenAI's "Branch in new chat" feature [2], show that even platform developers recognize the need for structured divergence. Yet these features remain tied to tabbed lists or sidebars, not visual knowledge.

Cognitive science offers an alternative. Larkin and Simon's foundational study on diagrammatic reasoning found that spatial representations reduce cognitive load by making relationships explicit [3]. Mind maps and concept diagrams allow learners to offload memory, visually cluster related ideas, and navigate complexity more intuitively [4]. Tools like Miro and Obsidian Canvas have popularized these methods in the productivity and UX world. MindMapper adapts this visual model to LLM interaction.

2 System Overview

MindMapper allows users to create prompts on a 2D canvas. Each prompt generates an answer node, connected visually. Users can fork off any bullet or idea in a response to create a new branch, which holds its own context.

Each chain (prompt → response → follow-up) forms a local reasoning thread. These branches can be revisited, collapsed, or compared side-by-side. Unlike traditional chats, MindMapper supports modular, navigable, and interpretable memory.

In addition to basic branching, each bullet point in a response node is interactive. Hovering over a bullet reveals three options:

- **Expand:** Instantly generates a follow-up answer node to that bullet using the current LLM context.
- **Sources:** Performs a web search and returns related articles or links in a dedicated "sources" node.
- **Custom Query:** Opens a query input box linked to the bullet. When the user submits, it creates a user-defined query node and its corresponding answer node.

Each query—whether a root, expansion, or custom follow-up—initiates a hybrid retrieval-generation pipeline. The system first calls the Perplexity Search API to fetch real-time, high-relevance web results. These results are then summarized by an LLM (e.g., GPT-5 or Claude Opus) into concise, factual bullet points displayed in the answer node. This design ensures every branch remains contextually coherent and grounded in the latest information, while the "Sources" option continues to display Perplexity's retrieved links directly.

Each node type—query, answer, and web sources—is color-coded for visual distinction and traceability across the map.

3 Memory and Summarization Strategy

LLMs operate within a finite context window. MindMapper avoids token overflow and context confusion through:

- **Branch Summaries:** Each branch maintains a running summary, generated and updated by summarization models (e.g., Claude or GPT-4o).
- **Vector Retrieval:** Every node and summary is embedded using sentence transformers and stored in a FAISS vector database [5].
- **Local Recall:** When a user expands a node, the system retrieves only the most relevant vectors from that branch. This enables branch-local context and minimizes drift.

This approach follows best practices from LangChain's vector-backed memory systems [6] and Pinecone's RAG pipelines [7].

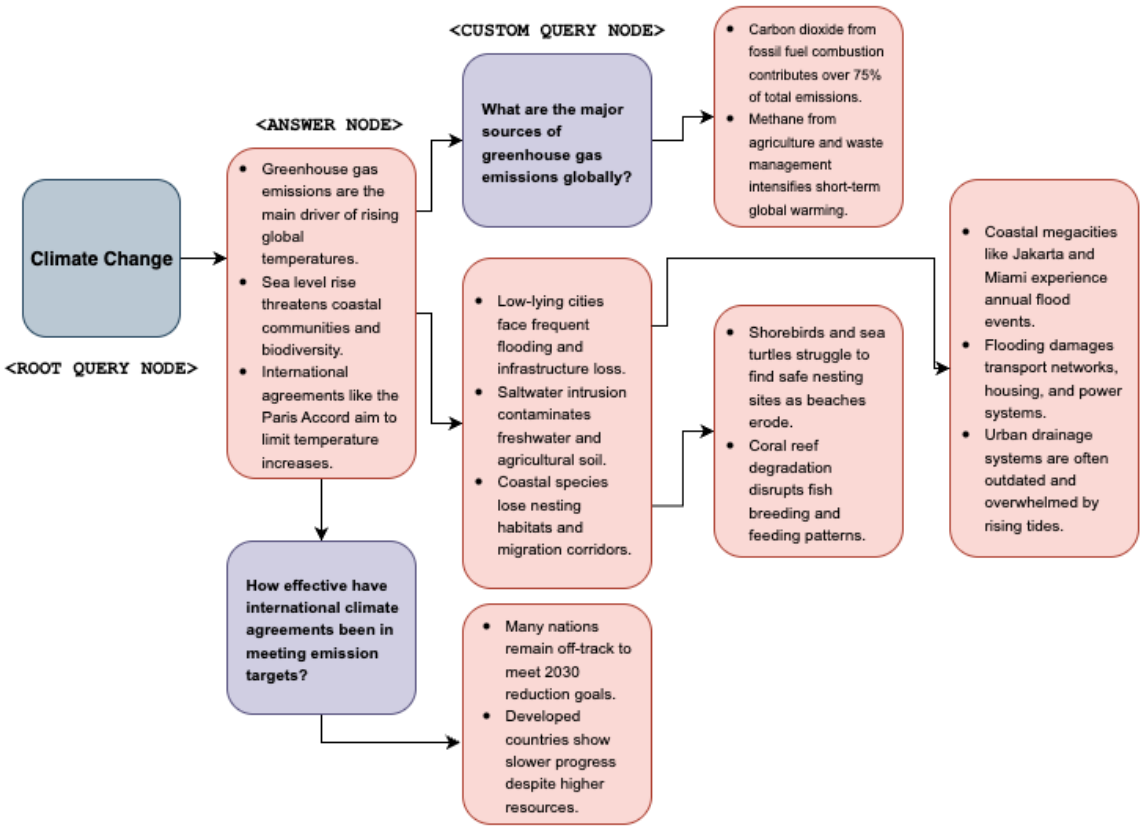

Figure 1: Preliminary sketch of the MindMapper interface showing query, answer, and sources nodes with branching connections.

4 Interface and UX Design

Inspired by real-world spatial tools, MindMapper provides:

- **Canvas Navigation:** Users pan, zoom, and group nodes spatially.
- **Expandable Nodes:** Clicking a bullet generates a new connected branch.
- **Mini-map Overview:** Large boards remain manageable with top-level map views.

Each bullet-level expansion is semantically linked to its originating text. Answer nodes are color-coded (e.g., blue), query nodes (e.g., green), and source nodes (e.g., orange), helping users distinguish between AI responses, their own custom prompts, and external content. This fine-grained control makes MindMapper suitable for research, exploration, and structured dialogue workflows.

References:

- **Obsidian Canvas** enables visual note linking [8].
- **Miro AI** offers sticky summarization and flowchart generation [11].
- **NotebookLM** recently introduced mind-map overviews for AI study sessions [9].
- **LLMapper** by Arango auto-generates concept maps from GPT-4 [10].

5 Implementation Stack

- **Frontend:** ReactJS with Konva.js for interactive canvas rendering and node manipulation. UI will support hover-triggered menus, color-coded nodes, zoom/pan, and a mini-map for navigation.
- **Backend:** FastAPI (Python) to handle LLM orchestration, Perplexity Search integration, vector retrieval, and summarization. Every node interaction (create, expand, or custom query) will trigger a retrieval step followed by an LLM summarization.
- **Memory:** FAISS vector index with sentence-transformer embeddings for local branch memory. Each query retrieves semantically similar prior nodes within its branch. Branch-level summaries are stored and updated incrementally.
- **LLMs:** GPT-5 (or Claude Opus) for summarizing retrieved web results into concise responses. This ensures each branch remains grounded in the most recent and factual context.
- **Web Search:** Perplexity API as the primary search engine for real-time retrieval. All root, expand, and custom queries first call this API; the "Sources" feature displays its results directly without summarization. Conditional fallbacks (e.g., SerpAPI) may be used as backups.
- **Persistence:** PostgreSQL to store boards, nodes, edges, metadata, and user sessions. Boards are autosaved and reloadable across sessions.

6 Evaluation Plan

A 2-condition within-subject user study will be conducted with 8-12 participants.

- **Conditions:** Standard chat UI vs. MindMapper.
- **Tasks:** Participants will explore a multi-topic reasoning task (e.g., designing a study, comparing models).
- **Metrics:** Navigation time, recall accuracy, satisfaction (SUS), and qualitative UX feedback.

7 Timeline (14 Weeks)

- **Week 1: Planning and Setup**
  Finalize project scope, core functionality, and evaluation goals. Define research questions related to multi-context LLM interfaces and interaction patterns. Set up development environment and select frontend (ReactJS + Konva), backend (FastAPI), and memory stack (FAISS + PostgreSQL).

- **Week 2: UI Mockups and Interaction Schema**
  Create detailed low-fidelity wireframes of canvas UI. Define node types (query, answer, source), branching behaviors, and hover interaction options ("Expand," "Sources," "Custom query"). Document data model schema for node representation and graph layout.

- **Week 3: Canvas Base and Node Rendering**
  Implement core canvas functionality using Konva.js. Enable node placement, dragging, line drawing between nodes, and basic panning/zooming controls. Support color-coded nodes with fixed spacing and positioning logic.

- **Week 4: Branching Logic and Hover Options**
  Enable branching from specific bullet points within response nodes. Add hover menus for each bullet offering three options: auto-expand, web search (sources), or custom query. Visualize linked child nodes accordingly.

- **Week 5: Retrieval–Summarization Pipeline Integration**
  Implement the hybrid pipeline where each query triggers a Perplexity Search for up-to-date web content, followed by summarization through GPT-5 or Claude Opus. Ensure consistent formatting of retrieved data, caching for repeated queries, and smooth integration into node rendering. The "Sources" function will continue to display Perplexity Search results directly.

- **Week 6: LLM Integration and Prompt Routing**
  Connect to OpenAI and/or Claude APIs. Route queries from each node to appropriate LLM backend and parse response. Format response into bullet-point layout. Handle context inheritance from parent branch.

- **Week 7: FAISS Memory System**
  Embed each node's content using a sentence transformer model. Store all embeddings in a FAISS index. On any new query, retrieve the top-N most similar vectors from the same branch path for summarization or memory injection.

- **Week 8: Branch Summarization and Memory Management**
  Implement summarization logic using a lightweight model (e.g., Gemini Flash or GPT-3.5). Periodically condense long branches into summary nodes. Inject these summaries instead of raw transcripts to preserve token space and improve model coherence.

- **Week 9: UX Refinements and Navigation Tools**
  Improve canvas polish, node animations, hover affordances, and font scaling. Add a minimap overview, branch collapse/expand controls, and breadcrumb tracing for nested threads. Support node hover previews.

- **Week 10: Onboarding and Persistence**
  Add onboarding walkthroughs for new users. Build save/load functionality using PostgreSQL to persist canvas state, branches, and interaction history across sessions.

- **Week 11: Internal QA and Pilot Testing**
  Perform internal QA and bug fixing across node logic, API calls, and memory retrievals. Conduct pilot tests with 2–3 peers to validate usability, memory scope, and performance under large boards.

- **Week 12: User Study Deployment**
  Recruit 12 participants. Run comparative study between MindMapper and standard LLM chat UI. Capture task completion time, navigation effort, system usability scale (SUS), and qualitative feedback.

- **Week 13: Data Analysis and Reporting**
  Analyze study data, compute metrics (e.g., average branching depth, recall accuracy, satisfaction). Visualize results and prepare charts, quotes, and findings summaries.

- **Week 14: Final Wrap-Up and Submission**
  Polish UI for demo, finalize documentation and codebase, and submit project report. Record and edit a demonstration video walking through system capabilities and use cases.

References

[1] Reddit. (2025). The Hidden UX Problem Killing Our LLM Conversations. *r/ClaudeAI*. `https://www.reddit.com/r/ClaudeAI/comments/1l2ze0n/the_hidden_ux_problem_killing_our_llm/`

[2] OpenAI. (2025, Sept 4). ChatGPT — Release notes: Branch conversations on web. https://help.openai.com/en/articles/6825453-chatgpt-release-notes

[3] Larkin, J. H., & Simon, H. A. (1987). Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1), 65–100. https://www.sciencedirect.com/science/article/pii/S0364021387800265

[4] Kefalis, C., Skordoulis, C., & Drigas, A. (2025). A Systematic Review of Mind Maps in STEM Education. *Computers*, 14(6), 204. https://www.mdpi.com/2073-431X/14/6/204

[5] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS 33*. https://arxiv.org/abs/2005.11401

[6] LangChain Documentation. (2024). Memory — concepts and APIs. https://python.langchain.com/docs/concepts/memory/

[7] Pinecone Documentation. (2025). Build a RAG chatbot (vector search + LLM + LangChain). https://docs.pinecone.io/guides/get-started/build-a-rag-chatbot

[8] Obsidian Help. (2025). Canvas (visual note-taking core plugin). https://help.obsidian.md/plugins/canvas

[9] Google Workspace Updates. (2025, Mar 19). New features available in NotebookLM: Mind Map. https://workspaceupdates.googleblog.com/2025/03/new-features-available-in-notebooklm.html

[10] Arango, J. (2024, Mar 17). LLMapper: An AI Concept Mapping Tool Using GPT-4. https://jarango.com/2024/03/17/llmapper-an-ai-concept-mapping-tool/

[11] Miro Help Center. (2025). Miro AI overview. https://help.miro.com/hc/en-us/articles/28765406244498-Miro-AI-overview