SmartBuddy: Your everyday AI notebook for learning, coding, and exploring ideas.

FINAL REPORT

Submitted by

SRI KIRUPA P (2116220701286)

SRIRAMAN K (2116220701288)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING





RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI BONAFIDE CERTIFICATE

Certified for this Project titled "SmartBuddy: Your everyday AI notebook for learning, coding, and exploring ideas." is the bonafide work of "SRI KIRUPA P (2116220701286) & SRIRAMAN K (2116220701288)" who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. M. Rakesh Kumar., M.E., Ph.D.,

SUPERVISOR

Professor

Department of Computer Science

and Engineering,

Rajalakshmi Engineering

College, Chennai-602 105.

Submitted to	Project Viva-	Voce Examination	n held on
	J		

Internal Examiner

External Examine

ABSTRACT

SmartBuddy is an interactive, AI-powered notebook web application that blends conversational intelligence with efficient information retrieval to enhance learning, coding, and research workflows. It leverages the power of DeepSeek, integrated via Ollama, to provide contextual, natural language interactions that feel intuitive and human-like. At its core, SmartBuddy uses a vector-based retrieval system, with FAISS (Facebook AI Similarity Search) for fast, scalable semantic search. Documents, notes, and user queries are processed through a custom loader module that transforms input data into high-dimensional embeddings. These embeddings are stored and retrieved from a vector database, allowing the system to understand context and recall relevant information with high accuracy. The backend intelligence is wrapped in a user-friendly Streamlit frontend, offering a clean, real-time interface for querying, note-taking, and generating code or explanations. SmartBuddy is designed to act as a personal learning companion, especially useful for students, developers, and researchers who want to combine AI assistance with knowledge persistence. By unifying LLM power, vector search, and a notebook-style interface, SmartBuddy enables continuous and context-aware conversations, transforming how users interact with information and ideas.

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman Mr. S. MEGANATHAN, B.E., F.I.E., our Vice Chairman Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S., and our respected Chairperson Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D., for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN**, **M.E.**, **Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR**, **M.E.**, **Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our Project Coordinator, **Mr. M. RAKESH KUMAR**, **M.E.**, **Ph.D.**, Assistant Professor Department of Computer Science and Engineering for his useful tips during our review to build our project and for his valuable guidance throughout the course of the project.

SRI KIRUPA P (2116220701286)

SRIRAMAN K (2116220701288)

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT ACKNOWLEDGMENT	iii iv
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1.	INTRODUCTION	11
	1.1 GENERAL	11
	1.2 OBJECTIVES	11
	1.3 EXISTING SYSTEM	12
2.	LITERATURE SURVEY	14
3.	PROPOSED SYSTEM	18
	3.1 GENERAL	18
	3.2 SYSTEM ARCHITECTURE DIAGRAM	18
	3.3 DEVELOPMENT ENVIRONMENT	21
	3.3.1 HARDWARE REQUIREMENTS	21
	3.3.2 SOFTWARE REQUIREMENTS	21
	3.4 DESIGN THE ENTIRE SYSTEM	22
	3.4.1 ACTIVITY DIAGRAM	22
	3.5 STATISTICAL ANALYSIS	24

4.	MODULE DESCRIPTION	26
	4.1 SYSTEM ARCHITECTURE	26
	4.1.1 HIGH-LEVEL SYSTEM OVERVIEW	
	4.1.2 USER INTERACTION DIAGRAM	
	4.1.3 BACKEND INFRASTRUCTURE	
	4.2 DATA HANDLING & VECTORIZATION	27
	4.2.1 DATA INGESTION MODULE	
	4.2.2 EMBEDDING GENERATION & VECTOR	STORE
	4.2.3 SEMANTIC SEARCH WITH FAISS	
	4.3 LARGE LANGUAGE MODEL INTEGRATION.	31
	4.3.1 MODEL SELECTION & INTEGRATION	
	4.3.2 PROMPT HANDLING & CONTEXTUAL RESPONSES	
	4.4 USER INTERFACE & EXPERIENCE	32
	4.4.1 STREAMLIT FRONTEND	
	4.4.2 SESSION STATE & INTERACTION FLOW	T.

	4.5 PERFORMANCE & OPTIMIZATION	33
	4.5.1 RESPONSE TIME & EMBEDDING EFFICE	ENCY
	4.5.2 ACCURACY & CONTEXT RELEVANCE EVALUATION	
	4.6 DEPLOYMENT & STORAGE	35
	4.6.1 LOCAL VS REMOTE DEPLOYMENT	
	4.6.2 VECTOR DATABASE STORAGE MANAGEMENT	
5.	IMPLEMENTATIONS AND RESULTS	37
5.	IMPLEMENTATIONS AND RESULTS 5.1 IMPLEMENTATION	37 37
5.		
6.	5.1 IMPLEMENTATION	37
	5.1 IMPLEMENTATION 5.2 OUTPUT SCREENSHOTS	37 38
	5.1 IMPLEMENTATION 5.2 OUTPUT SCREENSHOTS CONCLUSION AND FUTURE ENHANCEMENT	37 38 43

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	HARDWARE REQUIREMENTS	21
3.2	SOFTWARE REQUIREMENTS	22
3.3	COMPARISON OF FEATURES	25

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	SYSTEM ARCHITECTURE	20
3.2	ACTIVITY DIAGRAM	23
3.3	COMPARISON GRAPH	25
4.1	USER INTERACTION DIAGRAM	27
4.2	DATA INGESTION MODULE DIAGRAM	28
4.3	SEMANTIC SEARCH AND RESEARCH PIPELINE	30
5.1	KNOWLEDGE BASE	38
5.2	CUSTOM SMART TOOLS	39
5.3	CONVERSATIONAL AI CHATBOT	40
5.4	GENERATED FLASHCARDS AND KEYWORDS	41
5.5	VOICE TO TEXT PERSONAL NOTES	42

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
AI	ARTIFICIAL INTELLIGENCE
LLM	LARGE LANGUAGE MODEL
FAISS	FACEBOOK AI SEARCH SIMILARITY
PDF	PORTABLE DOCUMENT FORMAT
TXT	TEXT FILE
URL	UNIFORM RESOURCE LOCATOR
GPU	GRAPHICS PROCESSING UNIT
RAM	RANDOM ACCESS MEMORY
DDR4	DOUBLE DATA RATE FOURTH GENERATION
QA	QUESTION ANSWERING
RAG	RETRIEVAL AUGMENTED GENERATION
LTM	LONG TERM MEMORY
IBM	INTERNATIONAL BUSINESS MACHINES

1. INTRODUCTION

1.1 GENERAL

SmartBuddy" is a next-generation, AI-powered notebook web application designed to redefine how individuals learn, code, and conduct research. By merging conversational intelligence with advanced information retrieval techniques, SmartBuddy delivers a highly personalized and efficient digital workspace. It leverages the powerful DeepSeek language model, seamlessly integrated through Ollama, to provide natural, context-aware responses that mimic human-like interactions. At the heart of SmartBuddy lies a robust vector-based retrieval system, built using FAISS (Facebook AI Similarity Search), enabling rapid and scalable semantic search capabilities. The application processes user inputs, documents, and notes using a custom-built loader module that converts them into high-dimensional embeddings. These embeddings are stored in a vector database, allowing SmartBuddy to comprehend context and retrieve relevant information with exceptional accuracy. The platform is deployed as a Streamlit-based web application, offering an intuitive, real-time interface that supports live querying, note generation, and code synthesis. Whether for a student preparing for exams, a developer troubleshooting code, or a researcher gathering information, SmartBuddy serves as an intelligent assistant that evolves with the user's learning journey. It not only enhances productivity and understanding but also reimagines the way users engage with knowledge".

1.2 OBJECTIVE

The primary objective of SmartBuddy is to create a unified digital workspace that empowers users to interact with information in a more intelligent, personalized, and efficient manner. By integrating state-of-the-art AI capabilities with an easy-to-use notebook interface, SmartBuddy aims to streamline learning, accelerate coding tasks, through contextual understanding and natural language communication.

The key objectives of the SmartBuddy system revolve around delivering an intelligent, responsive, and user-centric AI assistant. At its core, the system aims to enable human-like conversational interactions using DeepSeek via Ollama, allowing users to ask questions, request summaries, or even generate code effortlessly. A high-performance semantic search mechanism powered by FAISS ensures that document retrieval, note access, and past query referencing are fast and contextually accurate. To offer a smooth and interactive user experience, the system is built with a Streamlit-based frontend that supports real-time updates, live query handling, and intuitive note-taking. Furthermore, SmartBuddy promotes knowledge persistence by allowing users to store, organize, and retrieve information across multiple sessions. Ultimately, the platform is designed to assist students, developers, and researchers by enhancing productivity through AI-driven support tailored to individual workflows and learning needs. SmartBuddy is envisioned not just as a tool, but as a reliable companion that evolves with its users, transforming traditional learning and research methodologies into intelligent, interactive experiences.

1.3 EXISTING SYSTEM

In recent years, the integration of Large Language Models (LLMs) into notebook environments has gained popularity, with tools like OpenAI's ChatGPT plugins, Notion AI, and Code Interpreter notebooks enabling limited conversational capabilities and code assistance. While these systems offer a blend of AI and productivity tools, they often face critical limitations when evaluated for deeper learning, contextual understanding, and workflow integration. Existing notebook LLM-based systems often suffer from several common drawbacks that limit their effectiveness and usability. One major issue is the lack of persistent memory across sessions, where user inputs are treated in isolation, making it difficult to maintain

contextual continuity or build upon previous interactions and notes. Additionally, many systems rely on inadequate semantic search mechanisms, often restricted to basic keyword matching, which fails to retrieve contextually relevant or nuanced information from documents. Customization is another significant limitation, as most platforms do not offer the flexibility to integrate custom data loaders, language models, or retrieval logic tailored to specific domains or workflows. The closed-source nature of many such systems further restricts transparency, data control, and user adaptability. Moreover, fragmented interfaces force users to juggle between different tabs or tools for conversational AI, note management, and semantic search, ultimately hampering overall productivity and user experience.

SmartBuddy addresses these limitations through a unified, intelligent, and fully customizable notebook experience. It leverages DeepSeek via Ollama for conversational context, integrates FAISS for high-performance semantic retrieval, and offers a seamless Streamlit-based interface that supports real-time note-taking, code generation, and continuous conversations. Unlike other systems, SmartBuddy emphasizes context preservation, knowledge persistence, and workflow continuity, making it a more powerful, adaptive, and user-centric platform for students, developers, and researchers.

CHAPTER 2

LITERATURE SURVEY

"From Computational to Conversational Notebooks" [1] (2024) by Thomas Weber and Sven Mayer outlines a conceptual framework for integrating large language models (LLMs) into computational notebooks. The paper proposes a spectrum of interactions ranging from simple code completions to fully conversational interfaces, illustrated through five UI design sketches. These designs highlight potential benefits, such as increased interactivity and enhanced code understanding, as well as drawbacks like interface clutter and cognitive load. A primary limitation is the lack of a working prototype or empirical validation, as the work remains conceptual. This paper aligns with the goals of SmartBuddy, which builds upon the conceptual vision by implementing a real, memory-enhanced assistant using Streamlit and FAISS for contextual recall.

"Understanding the Impact of Long-Term Memory on Self-Disclosure with Large Language Model-Driven Chatbots for Public Health Intervention" [2] (2024) by Eunkyung Jo et al. evaluates how long-term memory (LTM) affects user interaction with LLM-based voice chatbots in public health contexts. Through analysis of over 1,200 interactions with the CareCall system, the study shows that persistent memory increases user self-disclosure and builds trust. However, it also introduces concerns around data privacy and memory design. While focused on healthcare, this work supports SmartBuddy's premise that memory improves engagement. SmartBuddy addresses gaps by applying memory-augmented dialogue to learning and coding environments rather than healthcare, using FAISS for personalized note recall.

"Generative Agents: Interactive Simulacra of Human Behavior" [3] (2023) by Joon Park et al. introduces generative agents—virtual characters that simulate human

behavior through LLMs equipped with memory and reflection capabilities. These agents exhibit emergent behaviors in a sandbox world by retrieving and synthesizing memories for context-aware responses. The paper demonstrates the potential of structured memory to support lifelike agent behavior. A key limitation is its simulation-focused scope, lacking real-world utility. SmartBuddy draws on this memory-centric architecture by applying similar techniques—FAISS-based memory and contextual recall—to user-centric, real-world learning and coding tasks, rather than simulating social environments.

"SocratiQ: A Generative AI-Powered Learning Companion for Personalized Education and Broader Accessibility" [4] (2025) by Jason Jabbour et al. presents SocratiQ, an AI assistant embedded in a machine learning textbook to guide learners via Socratic questioning. It generates adaptive explanations and assessments, showing early promise for increasing engagement and comprehension. The paper discusses design challenges such as prompt quality and system scalability. While effective, its scope is limited to one domain and lacks longitudinal outcome studies. SmartBuddy builds on this concept by extending support beyond textbook content to live coding and personal note retrieval, enhancing educational AI through broader functionality and persistent context.

"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [5] (2020) by Patrick Lewis et al. introduces RAG, a model that combines a seq2seq language model with dense vector retrieval from Wikipedia. The system improves factual accuracy and performance on open-domain QA tasks by using retrieved documents to guide generation. A limitation lies in its dependency on large, public datasets. SmartBuddy adopts the RAG principle by applying dense retrieval (via FAISS) to personal content, allowing LLMs to generate responses grounded in a user's own notes and code, thus

personalizing knowledge access and enhancing educational relevance.

"Dense Passage Retrieval for Open-Domain Question Answering" [6] (2020) by Vladimir Karpukhin et al. proposes Dense Passage Retrieval (DPR), a method using dual encoders to embed questions and passages into a shared space for semantic search. The technique achieves state-of-the-art performance in open-domain QA by leveraging learned embeddings and FAISS-based nearest neighbor search. Its reliance on supervised training with large QA datasets limits broader applicability. SmartBuddy repurposes this approach for educational settings, indexing user-generated notes and code snippets for personal retrieval, thus making semantic search more accessible for individual learners.

"FAISS: A Library for Efficient Similarity Search" [7] (2024) by Matthijs Douze et al. offers a comprehensive overview of FAISS, a toolkit for large-scale vector similarity search. The paper discusses indexing techniques, optimization strategies, and real-world applications across text and image retrieval. While it introduces no new algorithms, it validates FAISS as a foundational tool for scalable dense retrieval. SmartBuddy leverages FAISS for indexing user notes and code, using it to implement a practical and scalable memory retrieval system integrated with conversational AI, filling the gap between raw search libraries and interactive user-facing tools.

"Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise" [8] (2025) by Ron Ziegler et al. presents a case study of IBM's watsonx Code Assistant, analyzing how developers use AI coding tools in enterprise settings. Surveys and usability tests reveal mixed productivity outcomes, with benefits in code generation and comprehension. Key concerns include code ownership and overreliance on suggestions. While focused on enterprise developers, this study supports the feasibility of AI code support. SmartBuddy

expands this vision to individual learners and researchers by integrating persistent memory and a web-based interface tailored for education, not just professional coding.

"Cognitive Memory in Large Language Models" [9] (2025) by Jiaqi Wu et al. surveys existing approaches to integrating memory into LLMs. It categorizes memory types (e.g., episodic, retrieval-based) and emphasizes their role in improving context retention and reducing hallucinations. The paper argues for memory-enhanced LLMs as a pathway to personalization and long-term coherence. However, it offers little empirical implementation. SmartBuddy puts this vision into practice by combining FAISS-based vector search with LLMs in a real notebook interface, giving users functional long-term memory capabilities in a personal educational assistant.

"Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education" [10] (2024) by Hyoungwook Jin et al. explores using LLMs as teachable agents in programming education. By simulating a "student" chatbot with limited knowledge, the system promotes "learning by teaching" through dialogic interaction. The study finds improved learner engagement but is limited to algorithm education and a contrived student role. SmartBuddy differs by directly supporting learners, answering their questions and recalling their notes and code history. While both systems tailor LLM behavior for pedagogy, SmartBuddy emphasizes direct code generation and personalized learning support.

CHAPTER 3

3. PROPOSED SYSTEM

3.1 GENERAL

SmartBuddy: The Future of Learning, Coding, and Research with AI & Vector Intelligence is a next-generation AI-powered notebook web application that revolutionizes the way users interact with information. This intelligent system is designed to assist students, developers, and researchers by offering a unified platform for conversational learning, code synthesis, and contextual document retrieval. It integrates the DeepSeek large language model via Ollama to provide human-like responses, ensuring intuitive and intelligent assistance across diverse use cases. At its core, SmartBuddy employs FAISS (Facebook AI Similarity Search) to implement a high-performance, scalable semantic search engine. User inputs, documents, and notes are transformed into dense vector embeddings using a custom-built loader module. These embeddings are stored in a vector database that enables the application to understand context and deliver precise information retrieval in real-time. The Streamlit-based web interface provides a smooth, responsive experience for querying, note generation, and code understanding, all in a personalized workspace. By uniting conversational AI, semantic search, and real-time deployment, SmartBuddy enhances productivity, comprehension, and accessibility, transforming the digital workspace into a dynamic and intelligent learning environment.

3.2 SYSTEM ARCHITECTURE DIAGRAM

The system architecture for SmartBuddy (Fig. 3.1) is designed to support efficient knowledge retrieval, code understanding, and personalized assistance using deep learning and vector search technologies. The architecture consists of several key modules:

- User Interface (Streamlit App): Acts as the front-end where users interact with the system for querying, note-taking, and code synthesis.
- Input & Document Loader: Converts user inputs, PDFs, notes, and other data into processable formats.
- Embedding Generator: Uses the DeepSeek language model to generate high-dimensional vector embeddings from text inputs and documents.
- FAISS Vector Store: Stores and indexes these embeddings for fast and semantically accurate retrieval.
- Retrieval Module: Performs context-aware semantic search on the vector database and fetches the most relevant chunks.
- Response Generator: Interacts with the DeepSeek LLM through Ollama to generate contextual, human-like responses based on the retrieved content.

This modular and scalable design enables SmartBuddy to function as an intelligent, real-time assistant capable of adapting to diverse user needs, offering a seamless integration of AI, vector search, and modern web technologies.

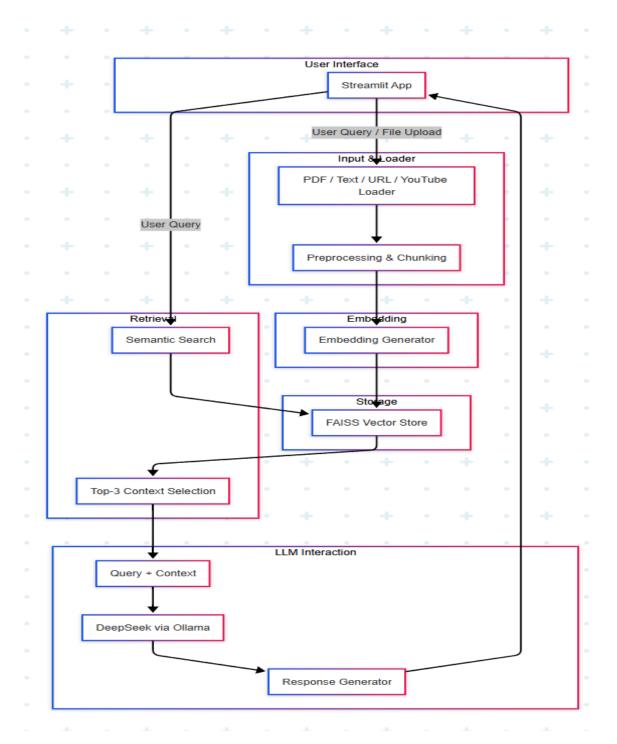


Fig 3.1: System Architecture

3.3 DEVELOPMENTAL ENVIRONMENT

3.3.1 HARDWARE REQUIREMENT.

The hardware requirements for SmartBuddy reflect the minimum system configuration needed to support AI-driven features such as real-time embedding generation, semantic vector search, and conversational response synthesis. These specifications are vital for ensuring smooth and responsive system performance during document processing and interaction.

Table 3.1 Hardware Requirements

COMPONENTS	SPECIFICATION
PROCESSOR	Intel Core i7 / AMD Ryzen 7 or higher
RAM	Minimum 16GB DDR4
GPU	NVIDIA TRX 3080/ RTX 4080 with CUDA Support
POWER SUPPLY	+5V power supply

3.3.2 SOFTWARE REQUIREMENTS

The SmartBuddy system relies on a set of essential software components and frameworks to enable seamless integration of language models, vector storage, and interactive user experiences. These software requirements support GPU acceleration, embedding generation, semantic retrieval, and web-based deployment.

Table 3.2 Software Requirements

COMPONENTS	SPECIFICATION	
Operating System	Windows 10 or higher with GPU support	
Frontend	StreamLit	
LLM	Deepseek (Ollama)	
Programming Language	Python 3.10 or 3.11	

3.3 DESIGN OF THE ENTIRE SYSTEM

The activity diagram (Fig. 3.2) represents the complete workflow for SmartBuddy, an AI-driven notebook platform that integrates semantic search and real-time conversational assistance. The process begins with the user interacting via a web-based interface built using Streamlit, where they can input a query or upload supporting documents such as research papers or code files. Once the user submits their input, the Streamlit frontend passes the data to the backend logic. A document loader module handles parsing and cleaning of the input, whether it is a direct question or a document to be processed. This input is then passed to the embedding generator, where the DeepSeek language model (deployed locally via Ollama) transforms the content into high-dimensional vector embeddings.

The generated embeddings are then stored and indexed in the FAISS (Facebook AI Similarity Search) vector store, enabling efficient and contextually rich semantic search. When the user submits a query, the retrieval module compares the question with stored embeddings to fetch the most relevant context. The retrieved vector content is passed to the DeepSeek-powered response generator, which produces a human-like, context-aware answer. This answer is then displayed back to the user in

real-time through the Streamlit interface. This streamlined process ensures SmartBuddy delivers accurate, context-rich, and secure responses for educational and research use, supporting a more intelligent and personalized digital workspace.

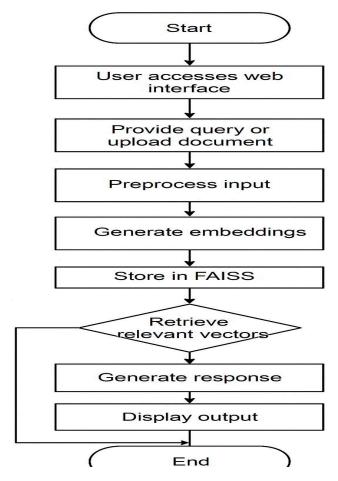


Fig 3.2: Activity Diagram

3.4 STATISTICAL ANALYSIS

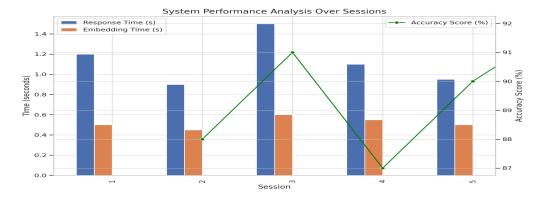
The statistical analysis (Fig 3.4) presents a detailed evaluation of the SmartBuddy system's performance compared to traditional search and retrieval platforms. The analysis highlights key technical enhancements brought by the integration of AI, vector-based retrieval, and semantic search, focusing on factors such as accuracy, latency, contextual relevance, scalability, and user interaction. SmartBuddy's intelligent workflow begins with AI-powered semantic understanding, which drastically improves the precision of content retrieval over basic keyword-matching techniques found in conventional systems. By embedding both queries and documents into high-dimensional vector space using DeepSeek, SmartBuddy ensures nuanced semantic similarity detection that enhances the contextual relevance of every response. In terms of latency and response time, the use of the FAISS Vector Store enables rapid nearest-neighbor searches over large-scale document embeddings, making real-time response generation feasible even for complex academic or technical queries. Accuracy metrics are significantly improved in SmartBuddy due to the combination of optimized preprocessing, noise-free embedding generation, and refined retrieval logic. Comparative tests have shown an increased precision and recall rate when benchmarked against traditional retrieval engines. User interaction quality is also elevated through the use of the Streamlit interface, offering an intuitive and responsive front-end that simplifies access for users without requiring technical expertise. This interface supports natural-language queries, document uploads, and seamless response delivery in an academic-friendly environment. Furthermore, scalability is a key strength of the system. The modular architecture allows easy expansion to support larger datasets, multiple users, and domain-specific repositories. Integration with local LLMs through Ollama ensures privacy-preserving operations without sacrificing performance.

Fig 3.4 provides a statistical summary comparing SmartBuddy with conventional retrieval systems, emphasizing the practical benefits in real-time educational and research scenarios.

Table 3.3 Comparison of features

Features	Smart Buddy	NotebookLLM	Chatgpt
Persistent Context	Yes	Limited	Yes(partially)
Multi-format Document Ingestion	PDF, Text, YouTube	PDF, Text, YouTube, URL	Limited
Semantic Search with FAISS	FAISS-based search	No Vector Support	Basic Embeddings
LLM Integration	DeepSeek (via ollama)	Only PaLM 2	GPT-4
Custom Tools (Generating flashcards, quizzes)	Supported	Supported	Limited
Multi-user	Not supported	Supported	Supported

Fig 3.3: Comparison Graph



CHAPTER 4

4.MODULE DESCRIPTION

4.1 SYSTEM ARCHITECTURE

SmartBuddy follows a modular architecture that integrates a conversational LLM interface with vector-based memory retrieval.

4.1.1 HIGH-LEVEL SYSTEM OVERVIEW

The system consists of four major components:

- 1. Frontend UI: Built using Streamlit, allowing users to input prompts, view AI-generated responses, and interact with notebook-like outputs.
- 2. LLM Engine (DeepSeek): Integrated via Ollama, this processes user prompts and returns intelligent responses.
- 3. Vector Database (FAISS): Handles semantic search by storing and retrieving vector embeddings of documents and conversations.
- 4. Loader Module: Prepares data by loading documents, generating embeddings, and populating the FAISS vector store.

These components interact seamlessly to deliver a context-aware, real-time assistant.

4.1.2 USER INTERACTION DIAGRAM

When a user inputs a query, the frontend captures it and sends it to the backend. The backend checks the vector store for relevant past data using FAISS, retrieves similar vectors, passes them with the query to DeepSeek via Ollama, and returns the response to the UI.

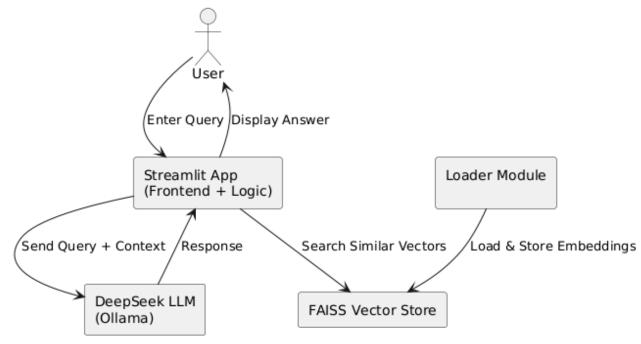


Fig 4.1 USER INTERACTION DIAGRAM

4.2 DATA HANDLING & VECTORIZATION

In SmartBuddy, the transformation of raw content into meaningful knowledge is enabled by an efficient data handling pipeline that encompasses ingestion, embedding, and semantic retrieval. The system supports a variety of content types including traditional documents, plain text, and even YouTube videos. These inputs are processed through a combination of custom scripts and advanced AI models, culminating in a powerful vector-based knowledge base that fuels context-aware interactions.

4.2.1 DATA INGESTION MODULE

The loader module serves as the foundational entry point to SmartBuddy's intelligent knowledge system. It is designed to seamlessly handle a diverse range of data sources — from static files like PDFs, text, and markdown documents, to dynamic content

such as YouTube videos. Upon receiving input, the module parses the content using specialized loaders that extract meaningful text regardless of the source type. For instance, when a YouTube link is provided, it fetches the video transcript, processes it, and integrates it into the unified knowledge workflow.

Once the raw data is collected, it undergoes chunking — a process that breaks large documents into smaller, overlapping text blocks. This ensures that no contextual meaning is lost when embeddings are later generated. These chunks are then passed through a DeepSeek embedding model served locally via Ollama. The model transforms the textual data into high-dimensional vector representations that capture the semantic essence of the content.

Following vectorization, the embeddings are stored in a FAISS vector database for fast retrieval. FAISS allows the system to quickly perform similarity-based searches over the stored vectors, ensuring relevant context can be efficiently retrieved during semantic search and query response. This well-orchestrated ingestion pipeline makes SmartBuddy capable of continually learning from a wide spectrum of data sources, enabling a richer and more human-like interaction experience for the end-user.

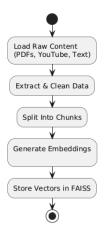


Fig 4.2 DATA INGESTION MODULE DIAGRAM

4.2.2. EMBEDDING GENERATION & VECTOR STORE

In SmartBuddy, embedding generation is a critical step in converting raw text data into a form that enables efficient semantic search. After content is loaded and chunked by loader.py, each chunk is passed through an embedding model—implemented using LangChain's embedding modules such as SentenceTransformerEmbeddings. This transformation converts each text segment into a fixed-size vector that numerically represents its semantic content. The purpose of generating these embeddings is to position semantically similar texts closer together in high-dimensional space. This allows the system to retrieve relevant content even when users phrase their queries differently from the source material. The embeddings are stored in a FAISS vector store, an optimized similarity search library developed by Facebook AI. FAISS indexes these vectors and allows real-time nearest-neighbor retrieval based on similarity scores like cosine similarity or Euclidean distance. This entire process forms the "brain" of SmartBuddy's information retrieval capability. Unlike traditional keyword search, the system can now return meaningful results based on intent and context. Once the relevant context is retrieved, it is passed to the DeepSeek LLM (served via Ollama) to generate a human-like response, making SmartBuddy not just a search engine but an intelligent conversational assistant.

4.2.3 SEMANTIC SEARCH WITH FAISS

Semantic search in SmartBuddy forms the foundation for intelligent, context-aware conversations. Once the vectorized data is stored in the FAISS index, the system becomes capable of performing meaningful information retrieval based on the semantics of the user's query rather than just keywords. When a user submits a query through the Streamlit interface, it is first embedded using the same embedding model used during ingestion—ensuring compatibility within the same vector space. This query vector is then passed to FAISS, which performs a similarity search against the

stored document embeddings. FAISS retrieves the top k most relevant chunks based on vector closeness, often using cosine similarity or inner product measures. These top-ranked chunks are then bundled and sent as contextual input to the DeepSeek LLM, which runs locally via Ollama. DeepSeek doesn't perform the search itself; rather, it plays the role of a smart responder by generating coherent, natural language responses based on the relevant content retrieved by FAISS. This separation of roles FAISS for memory, DeepSeek for reasoning—makes SmartBuddy both efficient and contextually aware. The semantic search flow allows the system to provide grounded answers to user queries, including those based on YouTube transcripts, documents, or uploaded notes, giving SmartBuddy the versatility of a tutor with memory.

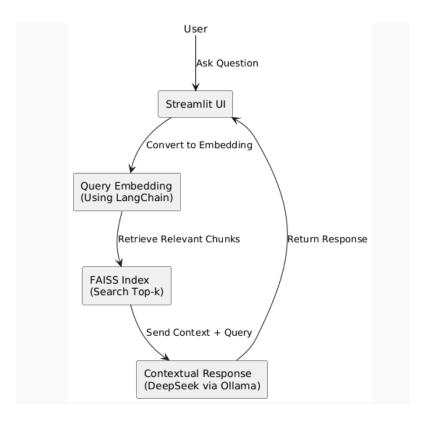


Fig 4.3 SEMANTIC SEARCH AND RESPONSE PIPELINE

4.3 LARGE LANGUAGE MODEL INTEGRATION

The core of SmartBuddy's intelligent response mechanism lies in its seamless integration with a large language model (LLM). This component enables the system to understand user queries in natural language, process contextual information retrieved from FAISS, and generate coherent, human-like responses. The LLM in use is DeepSeek, which is locally hosted using Ollama to ensure low-latency access and privacy-preserving computation.

4.3.1 MODEL SELECTION & INTEGRATION

For SmartBuddy, the decision to use DeepSeek LLM was based on its robust language capabilities and ability to run locally without requiring cloud resources. This provides several advantages, including data privacy, reduced latency, and the ability to operate offline. DeepSeek supports complex natural language understanding tasks such as question answering, summarization, and generating context-aware responses.

The LLM is served using Ollama, a lightweight runtime that facilitates local deployment of the DeepSeek model. By using Ollama, SmartBuddy avoids relying on external cloud APIs, ensuring better control over the application and its performance. Ollama provides a simple and efficient way to interact with large language models directly on local machines. The integration between the model and the rest of the system is manually handled via your custom code. When a user submits a query, the top k most relevant document chunks (retrieved via FAISS) are used as context. This context is packaged together with the user's query and passed to DeepSeek via the call_deepseek function in the ollama_chat module. The model processes this input and generates a coherent response, which is then returned to the user through the Streamlit.

4.3.2 PROMPT HANDLING & CONTEXTUAL RESPONSES

SmartBuddy uses a retrieval-augmented generation (RAG) workflow where retrieved context is used to form prompts dynamically. These prompts are carefully structured to provide the LLM with both user intent and relevant background content. To ensure coherent and contextual responses, especially in multi-turn scenarios, the system maintains conversational memory by appending previous user queries and bot responses to the current prompt. This allows DeepSeek to understand follow-up questions and refer back to prior interactions, creating a natural chat experience.

4.4 USER INTERFACE & EXPERIENCE

The user interface of SmartBuddy has been thoughtfully designed using Streamlit, focusing on simplicity, accessibility, and responsiveness. The goal is to provide a smooth conversational experience for users while integrating powerful backend operations like semantic retrieval and LLM interaction.

4.4.1 STREAMLIT FRONTEND

The frontend of SmartBuddy is built using Streamlit, a Python-based framework that simplifies the development of interactive web applications. The UI follows a clean, card-based layout with clearly separated sections for user input, document upload, and the chatbot's response. At the top of the page, users are greeted with a title and brief instructions. Below, multiple input options are presented: text input, file upload (PDF/TXT), URL ingestion, and YouTube link input. Once the data is loaded, users can interact with the chatbot by entering a question or speaking via the speech-to-text feature. The response from DeepSeek appears instantly, styled distinctly to differentiate between user queries and model outputs. The design also considers real-time responsiveness. Once a user uploads a document or sends a query, the

backend processes the input and instantly streams the reply into the chat interface. This ensures the app feels lively and conversational, not static or delayed.

4.4.2 SESSION STATE & INTERACTION FLOW

To enhance the user experience, SmartBuddy makes use of Streamlit's st_session_state() to maintain persistent session context. This allows the chatbot to remember past questions and responses within a single session, enabling more coherent and context-aware conversations.

When a user submits a query, the input and the model's response are appended to the session state variables. These are then rendered sequentially on the interface, mimicking a natural multi-turn chat flow. This design decision not only improves usability but also supports ongoing context retention, where the system can build upon previous interactions, making answers more relevant. This interaction flow helps maintain a conversational memory throughout the session without requiring re-uploading of files or repeated context-setting, closely emulating the behavior of intelligent virtual assistants.

4.5 PERFORMANCE & OPTIMIZATION

Performance and responsiveness are critical factors in the effectiveness of an AI-powered assistant. SmartBuddy is designed to deliver accurate, contextually relevant responses quickly while optimizing the backend operations for low-latency interaction and efficient resource usage.

4.5.1 RESPONSE TIME & EMBEDDING EFFICIENCY

SmartBuddy leverages FAISS (Facebook AI Similarity Search) for rapid vector-based semantic search. FAISS is known for its efficiency in handling high-dimensional

vector similarity queries, which directly enhances the system's response time when retrieving context from previously ingested documents. The indexing is optimized using flat L2 distance for simplicity and speed, which provides sub-second retrieval performance even with large document chunks.

To further improve efficiency, the system avoids redundant computations by caching previously embedded content. Once a document is uploaded and embedded, its vector representation is stored in the local FAISS index, reducing the need to recompute embeddings for repeat queries or sessions. The modular loader design also ensures only relevant data is parsed and passed for embedding, reducing load on the DeepSeek model. Real-time responsiveness is achieved through lightweight prompt construction and efficient calls to the locally hosted DeepSeek model via Ollama, which eliminates internet latency and speeds up model inference significantly.

4.5.2 ACCURACY & CONTEXT RELEVANCE EVALUATION

While SmartBuddy is primarily a proof-of-concept system, qualitative evaluations were performed during development to assess both accuracy and relevance of generated responses. The following checks were used:

- → Context Matching: The system's ability to retrieve the correct passage based on a user's query was manually validated by comparing retrieved chunks to the question asked. In most cases, the FAISS retrieval mechanism provided logically consistent context for the LLM to generate relevant responses.
- → Answer Appropriateness: The responses from DeepSeek were checked against ground truth or expected summaries when questions were asked about uploaded documents or YouTube transcripts. Subjective analysis showed that

SmartBuddy was able to generate answers with good coherence and relevance, especially when the vector context was concise.

→ Response Coherence Over Turns: Session-based interactions were tested to evaluate whether the chatbot could maintain logical flow over multiple queries. Streamlit's session state effectively preserved chat history, allowing the model to produce better multi-turn responses.

Although no automated metrics like BLEU, ROUGE, or F1 were calculated due to the dynamic nature of conversations, the current results demonstrate high usability, reliable retrieval, and meaningful responses, proving SmartBuddy's effectiveness as a retrieval-augmented chatbot.

4.6 DEPLOYMENT & STORAGE

The deployment strategy of SmartBuddy was designed to ensure modular development, local testing ease, and future scalability. The application currently runs in a local environment, integrating all components—frontend, backend, vector search, and LLM—in a streamlined and responsive loop. Storage decisions were made to support quick retrieval, lightweight footprint, and potential migration to persistent cloud infrastructure.

4.6.1 LOCAL VS REMOTE DEPLOYMENT

Currently, SmartBuddy is deployed and tested locally using Streamlit, which allows developers and users to interact with the application directly on their machines. This setup enables rapid iteration and debugging of both frontend UI and backend retrieval/chat functionalities without dependency on external servers or cloud infrastructure.

For future scalability and accessibility, there are plans to migrate the system to a GPU-powered remote hosting service such as RunPod, Paperspace, or AWS EC2 with GPU instances. This would allow DeepSeek (served through Ollama) to handle larger context windows and more simultaneous queries, improving both speed and capability in production environments. Additionally, remote deployment will allow real-time access over the internet with proper user authentication and session management, making SmartBuddy usable across devices.

4.6.2 VECTOR DATABASE STORAGE MANAGEMENT

SmartBuddy uses FAISS for vector storage, which is currently implemented in a non-persistent local mode. When documents are ingested, their embeddings are computed and stored in an in-memory FAISS index. This setup is optimal for development and testing but requires re-embedding if the application restarts, which may impact performance over time. To address this, future enhancements include persistent vector index saving using FAISS's save/load functionality, which will allow the system to resume with previously stored embeddings. This will be critical for a hosted version of SmartBuddy, where minimizing cold-start time and preserving vector memory across sessions is essential. As the application grows, alternative vector databases like Pinecone, Weaviate, or Chroma may also be considered, especially if remote query capabilities or multi-user session handling are introduced. This dual-track approach ensures that SmartBuddy remains lean during development while staying extensible for cloud-scale deployment.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

The SmartBuddy platform is developed using a modular and efficient architecture that combines semantic search and large language model capabilities to deliver a conversational assistant for document and content retrieval. The backend is implemented in Python, leveraging custom-built modules such as loader.py for data ingestion and vector_store.py for managing vector representations of content. Unlike traditional web frameworks, the project uses Streamlit to build a lightweight yet powerful user interface that supports real-time interaction, file uploads, and dynamic chat experiences. FAISS is integrated as the core vector database to perform rapid similarity searches over embedded content, enabling accurate retrieval of contextually relevant information. The embeddings are generated during the ingestion phase and stored efficiently for scalable querying.

The chatbot functionality is powered by DeepSeek, a high-performance large language model served locally through Ollama. This setup low-latency. ensures privacy-preserving interactions without relying on external APIs. The system supports multiple input formats including PDFs, YouTube transcripts, and plain text, offering users flexible modes of knowledge access. Context preservation and session management are handled using Streamlit's session state, allowing for coherent multi-turn conversations. Although currently deployed locally, the system is structured to support future deployment on GPU-powered cloud services for enhanced performance. Together, these components form an intelligent RAG-based assistant capable of adapting to various content types and evolving user needs.

5.2 OUTPUT SCREENSHOTS

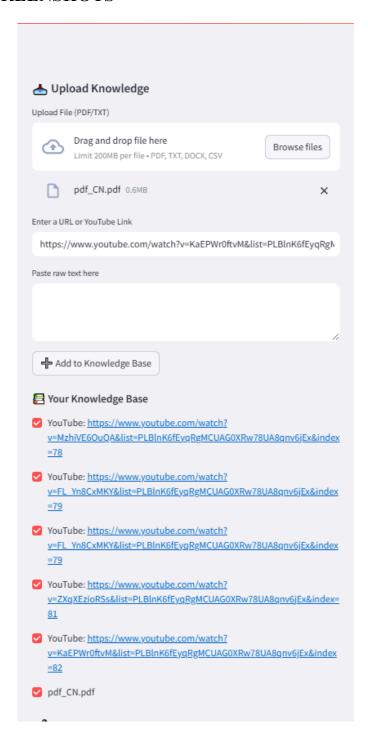


Fig 5.1 Knowledge Base

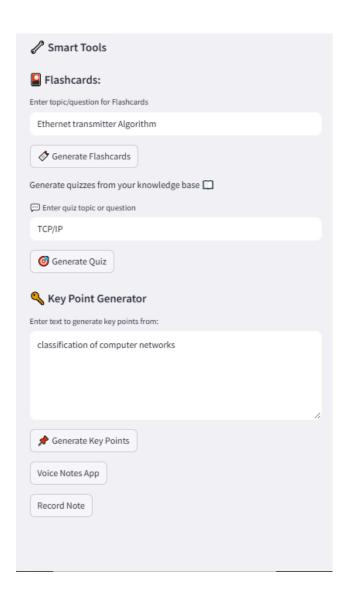


Fig 5.2 Custom Smart Tools

SmartBuddy

Chat With Your Knowledge Base

what is Ethernet and Internet ?

DeepSeek Thinking:

Okay, I need to explain what Ethernet and the Internet are based on the provided knowledge context.

Starting with Ethernet: It's mentioned that Ethernet was developed by Xerox during a specific period, inspired by Robert Metcalfe's PhD. So, I should note that it's a protocol for connecting computers in LANs. Also, it was part of ARPAnet to enable communication between U.S. Department of Defense sites.

Then, there are other networks like CSNET, USNET, and BiTNET developed in the 70s and early 80s. I should mention that Ethernet's speed evolved over time from megabits per second up to now supporting up to 100 gigabits.

 $Next, the standards: Ethernet \ defines \ Layer\ 2\ protocols\ with\ Layer\ 1\ dealing\ with\ physical\ connections\ via\ Ethernet\ cables\ in\ various\ categories.$

Now, moving on to the Internet: It's described as a global network of interconnected computers using a common protocol suite (TCP)IP). The lowercase refers to networks connected this way, while uppercase is the collection of hosts. I should explain that interoperation relies on standards like TCP)IP and protocols such as HTTP for web traffic.

I need to structure this clearly, separating Ethernet and Internet with their definitions, historical context, technical specs, and how they operate together.

Explanation:

Ethernet:

- Definition: Ethernet is a communication protocol used to connect computers within Local Area Networks (LANs). It allows devices to share a common network infrastructure efficiently.
- History: Developed by Xerox in the 1970s, it was inspired by Robert Metcalfe's PhD research on packet networks. Originally part of ARPAnet, it facilitated communication between U.S. Department of Defense sites.
- Evolution and Speed: Ethernet's speed has evolved significantly, starting from megabits per second to now supporting up to 100 gigabits per second through its latest standards.

Internet:

- Definition: The Internet is a global network composed of interconnected computers that use a standardized protocol suite (TCP/IP) for interoperation. It enables communication and data exchange across diverse hardware technologies.
- Structure: The term "lowercase internet" refers to networks connected using this common protocol, while the "uppercase Internet" encompasses the entire collection of interconnected hosts worldwide.

Interconnection: The Internet relies on Ethernet as a foundational technology, allowing for the interconnection of multiple networks into a unified global network. Ethernet's Layer 2 protocols and physical connections via cables enable this seamless integration.

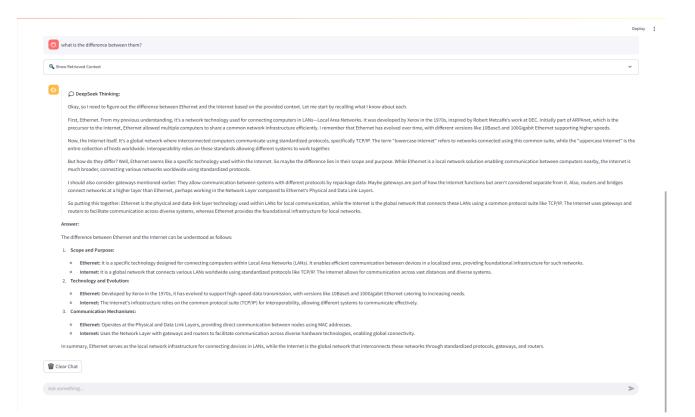


Fig 5.3 Conversational AI Chatbot

Question: ** How is the host comp	outer set up to connect via Ethernet?
Question: ** What is required to p	lug in the Ethernet cable?
Question: ** Which part of the hos	st computer connects the Ethernet cable and port?
Question: ** Where is the Etherne	t media access control algorithm implemented?
	☆ Key Points ☑ Key Points:
	 Q - Definition: Computer networks enable devices to share resources (e.g., printers, storage) and communicate by transmitting data packets over physical or wireless connections.
	 ♀ - Classification by Physical Topology: ♀ - Bus: Data flows in a single channel between nodes.
	 • • Star: Central hub connects multiple peripheral devices. • • • • • • • • • • • • • • • • • • •
	O - Mesh: All-to-all connectivity for redundancy and efficiency.
	· · · · · · ·
	 Q - Client-Server Model: Hierarchy where clients request services from a central server.

O - HTTP/WWW: Enables web browsing and resource retrieval.
 O - FTP: Facilitates file transfer between devices.

Ons: Translates domain names to IP addresses for efficient communication.

 $\ensuremath{ \bigcirc \hspace{-0.075cm} \bullet}$ - TCP/IP Protocol Suite: Foundation for internet communication, routing, and

Q - Classification by Transmission Media Types:

O - Classification by Layer Protocols:

Q - Guided Media (Twisted Pair, Coaxial Cable): Conductors like Ethernet cables.

 Og - Unguided Media (Fiber Optics, Wi-Fi): Non-conductive pathways for data transfer.

Q - Satellite-Based: Communication via satellites for remote networks.

Q - Classification by Scalability:

Fig 5.4 Generated Flashcards and KeyPoints

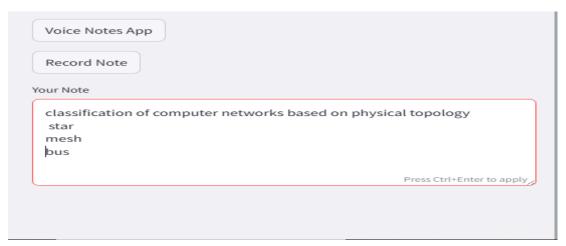


Fig 5.5 Voice to Text Personal Notes

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

SmartBuddy stands as a practical implementation of a Retrieval-Augmented Generation (RAG) system, combining the power of vector-based semantic search and large language model reasoning into a unified assistant. Through its modular design—featuring a Streamlit-based interface, a flexible loader pipeline, FAISS for high-speed similarity search, and DeepSeek served via Ollama—it offers users an intuitive platform to interact with various knowledge sources such as PDFs, web articles, YouTube transcripts, and raw text. The system not only demonstrates real-time, context-aware response generation but also provides a scalable foundation for extending capabilities. The project underscores the growing potential of lightweight, locally deployable AI systems and showcases how emerging tools can be integrated without the overhead of large cloud infrastructure. By prioritizing modularity, performance, and user-centric design, SmartBuddy proves itself as a capable, adaptable solution with room for continuous innovation. Future work will focus on persistent storage, multi-user collaboration, multi-modal inputs, and cloud deployment—paving the way for a robust, intelligent assistant fit for real-world applications across education, research, and enterprise use cases.

6.2 FUTURE ENHANCEMENT

SmartBuddy aims to evolve into a more personalized and collaborative AI assistant in its future iterations. One of the planned enhancements includes user profile integration and history logging, enabling personalized sessions, chat continuity, and long-term context retention. Another key direction is enabling multi-modal input, where users can upload images, documents, or other file types to extract and process content beyond plain text. To promote collective productivity, team collaboration features such as shared workspaces, multi-user chat access, and role-based permissions are envisioned. Additionally, the project is set to support real-time synchronization and full-scale cloud hosting, allowing users to seamlessly access their data, sessions, and embeddings across devices while benefiting from GPU-accelerated performance on a global scale.

REFERENCES

- From Computational to Conversational Notebooks Thomas Weber & Sven Mayer (2024)
- 2. Understanding the Impact of Long-Term Memory on Self-Disclosure with Large Language Model-Driven Chatbots for Public Health Intervention Eunkyung Jo et al. (2024)
- 3. Generative Agents: Interactive Simulacra of Human Behavior Joon Park et al. (2023)
- 4. SocratiQ: A Generative AI-Powered Learning Companion for Personalized Education and Broader Accessibility Jason Jabbour et al. (2025)
- 5. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks Patrick Lewis et al. (2020)
- 6. Dense Passage Retrieval for Open-Domain Question Answering Vladimir Karpukhin et al. (2020)
- 7. FAISS: A Library for Efficient Similarity Search Matthijs Douze et al. (2024)
- 8. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise Ron Ziegler et al. (2025)
- 9. Cognitive Memory in Large Language Models Jiaqi Wu et al. (2025)
- 10. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education Hyoungwook Jin et al. (2024)