Team Details

Team Leader

- Name: Abhinav Patel
- Roll No.: 240001004
- GitHub: `https://github.com/AbhinavPatel271`
- LinkedIn: `https://www.linkedin.com/in/abhinav-patel-275719330/`
- Skills: Python, C++, JavaScript, ReactJS, ExpressJS, Flask, PyTorch, Transformers, Generative AI, PostgreSQL, Git, Docker

Other Team Members

- **Karanam Venkata Lakshmi Sarath Chandra**
  - Roll No.: 240001039
  - GitHub: `https://github.com/Sarathchandra-kvl`
  - LinkedIn: NA
  - Skills: C++, Python, Deep Learning, NLP, PyTorch, Pandas, Scikit-learn, LangChain
- **Devanshu Dubey**
  - Roll No.: 240001024
  - GitHub: `https://github.com/DevanshuDubey`
  - LinkedIn: `https://www.linkedin.com/in/devanshuddubey/`
  - Skills: : C++, Kotlin, Python, PyTorch, Pandas, Scikit-Learn, ML, NLP, LLMs, Computer Vision, RAG, LangChain, Agents, Git
- **Anurag Prasad**
  - Roll No.: 240001011
  - GitHub: `https://github.com/LegendXAnurag`
  - LinkedIn: `https://www.linkedin.com/in/anurag-prasad-8279aa315/`
  - Skills: C++, Python, Deep Learning, NLP, HTML, CSS, JavaScript, ReactJS, NodeJS, ExpressJS
- **Saransh Halwai**
  - Roll No.: 240001066
  - GitHub: `https://github.com/saranshhalwai`
  - LinkedIn: `https://www.linkedin.com/in/saransh-halwai-478346171/`
  - Skills: Python, PyTorch, ML, DL, Diffusers, LangChain, Ollama, Multi-agent frameworks, Git, GitHub, Docker, NLP, LLMs, Whisper, Linux, MCP
- **Ankur Singh**
  - Roll No.: 240001009
  - GitHub: `https://github.com/Ankur556`
  - LinkedIn: `https://www.linkedin.com/in/ankur-singh-522331340/`
  - Skills: Neural Networks, LLMs, CSS, HTML

0.1 Project Overview

IITI-BOT is an intelligent chatbot designed to serve as an accurate information resource for students, faculty, and visitors by accessing a comprehensive knowledge base covering all aspects of IIT Indore. The system allows users to search for any information related to IIT Indore, enriching users' understanding of the institution.

This project aims to design and implement an intelligent chatbot using Pathway's Agentic RAG system that can accurately and autonomously answer any query related to IIT Indore. The system will utilize official institutional data and dynamically reason across documents to deliver context-rich responses.

0.2 Project Solution

The solution comprises a responsive, web-based chatbot application compatible with all devices. The application will be built using React for functionality and styled with Tailwind CSS for a modern, responsive design. It will include a dedicated admin section secured via Firebase Authentication or Google OAuth 2.0 for access control and new document uploads. The frontend will be deployed using platforms like Netlify or Vercel, ensuring easy, one-click access for users.

We propose to develop the chatbot as an Agentic Retrieval-Augmented Generation (RAG) architecture from scratch, leveraging Pathway's real-time data streaming pipeline and native RAG components without relying on any external multi-agent frameworks.

The chatbot will have two main sections of work:

1. Data Streaming Pipeline
2. Agentic Retrieval-Augmented Generation Pipeline

Following these, deployment work will be undertaken.

0.2.1 Pathway Data Streaming Pipeline

A containerized environment using Docker will be implemented for real-time data streaming with Pathway.

Initial Development Phase

During the initial development phase:

- We'll cache Pathway's persistent state locally using `pw.persistence.Backend.filesystem("./localPath")`. Later, we'll switch to `pw.persistence.Backend.s3(...)` with an AWS S3 bucket to ensure state consistency across team members and support scalable document growth.
- Documents (PDFs) will initially be stored locally and connected via `pw.io.fs.read()` in static mode for easier debugging. As we progress, we'll transition to streaming mode, move PDFs to AWS S3 bucket, and connect using `pw.io.s3()`. Admins will upload files through the web interface to the bucket via S3's pre-signed URLs generated by the backend.

Components of the Data-Streaming Pipeline

Parser Strategy Pathway provides a range of built-in parsers, including `Utf8Parser`, `UnstructuredParser`, `DoclingParser`, `PypdfParser`, `ImageParser`, and `SlideParser`. For this project, we have chosen to utilize the `DoclingParser`.

This strategy is particularly well-suited for our use case, as it enables table summarization directly from PDFs without requiring a vision-language model (VLM). The `DoclingParser` employs OCR through the Docling library to extract table content and convert it into Markdown format for further processing.

To activate this functionality, the `table_parsing_strategy` is set to "docling". The parser class wraps around Docling's `DocumentConverter` and includes support for parsing images using vision LLMs, offering flexibility for potential upgrades should resources become available in the future.

Text Splitter Strategy Pathway offers four options: `BaseSplitter`, `RecursiveSplitter`, `TokenCountSplitter`, and `NullSplitter`.

We will use `RecursiveSplitter`, which chunks text based on a hierarchy of separators, and its `chunk_overlap` parameter is particularly useful for preserving context.

Retrieval Factory Strategy Pathway offers the following retrieval options:

- Vector-Based Retrieval
- Non-Vector Retrieval
- Hybrid Retrieval

It includes multiple retrieval factory methods such as `BruteForceKnnFactory`, `UsearchKnnFactory` , `LshKnnFactory`, `TantivyBM25Factory`, and `HybridIndexFactory`.

We'll evaluate these methods, examine the mechanism of each retrieval approach, and determine which retrieval factory method is best suited. We'll aim to make the best use of Pathway's functions to filter out the relevant documents based on the query.

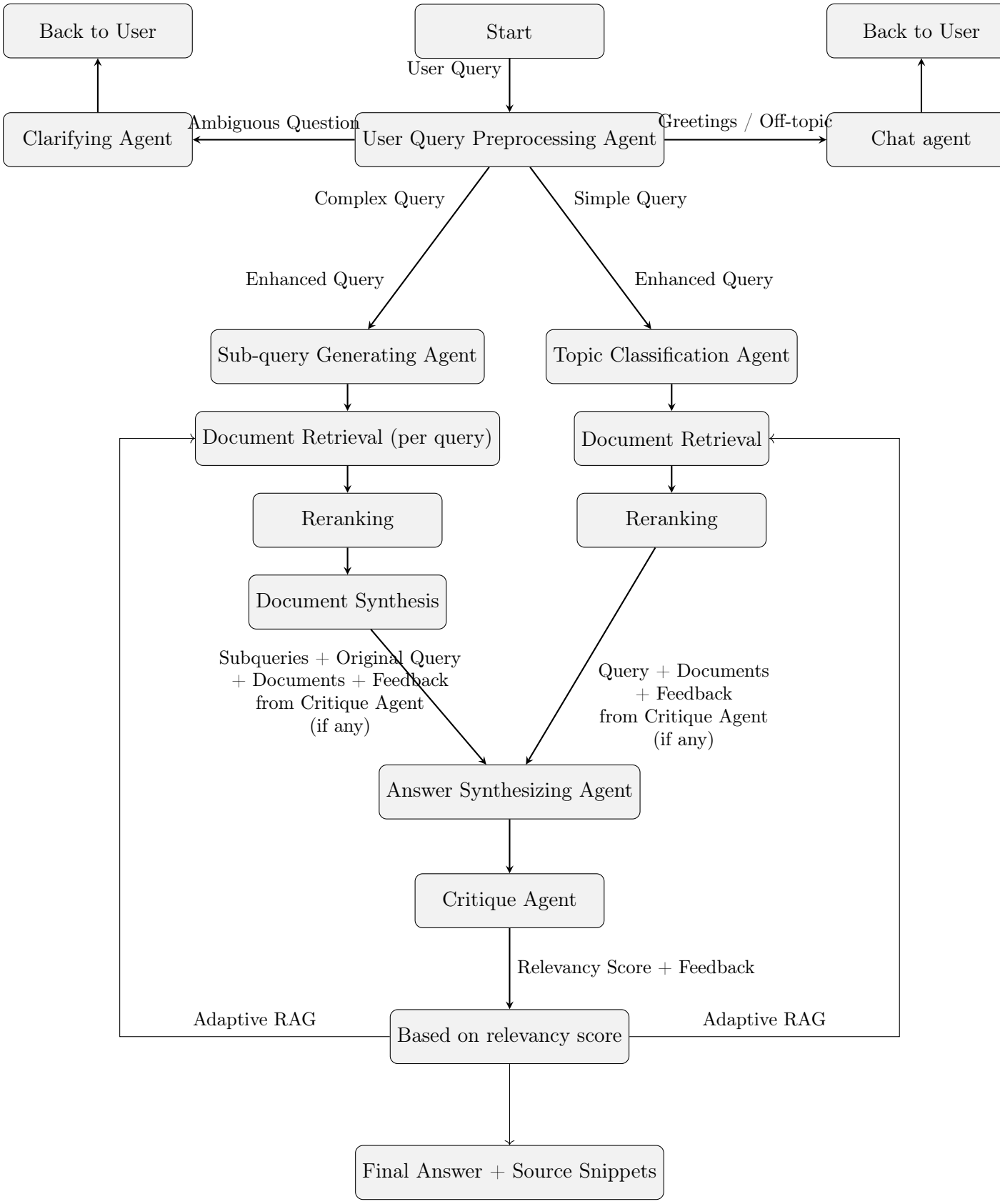
Embeddings Pathway's retrieval factories utilize embedders to generate vector representations for effective document retrieval.

Pathway offers the following embedders:

- OpenAIEmbedder
- LiteLLMEmbedder
- SentenceTransformers Embedder
- GeminiEmbedder

We are considering the use of `LiteLLMEmbedder` as it is highly compatible with Pathway and provides access to a wide range of embedding models to choose from.

0.2.2 Agentic Retrieval-Augmented Generation (RAG) Pipeline



The flowchart shown in the Figure illustrates our basic Agentic RAG pipeline. This serves as a starting point, and we plan to make improvements as we progress.

The user query preprocessing agent acts as a router agent and forwards the enhanced version of the query to the appropriate next agent based on the query type.

Agent Types

- **Clarifying Agent:** Engages with the user to clarify the query and eliminate any ambiguity.
- **Chat Agent:** Handles off-topic queries and greeting-related interactions.
- **Topic and Keywords Identifying Agent:** For simple, single-topic queries, it extracts relevant keywords to utilize the `meta_data` filter provided by Pathway's Document Store.
- **Sub-query Generating Agent:** For complex, multi-topic queries, it breaks the main query into multiple sub-queries—each focused on a distinct topic.

Retrieval (Using CRAG's Algorithm)

For document retrieval, we will use Pathway's provided retrieval factories, as outlined in the components of the data streaming pipeline.

Rerankers

Using the Pathway framework, we can rerank the retrieved documents. Pathway offers several rerankers:

- LLMReranker
- EncoderReranker
- CrossEncoderReranker

We plan to use the `CrossEncoderReranker` because it applies a sigmoid function that scores relevancy between 0 and 1, enabling precise comparison of retrieved documents' relevance. We will experiment with different models available within the `CrossEncoderReranker` to identify the best option. Additionally, we can filter the top-k documents using the `rerank_topk_filter` method. This mechanism functions similarly to the decompose-then-recompose algorithm in Corrective RAG, effectively filtering and reranking documents.

Document Synthesizer Agent

The retrieved documents are synthesized into a final output, with the synthesis weighted according to each document's importance as determined by the rerankers. To achieve this, we may utilize Pathway's `BaseContextProcessor` class, which formats the documents appropriately for the LLM context.

Augmentation

The query and the output from the Document Synthesizer are passed to the Answer Synthesizer Agent. For this purpose, we will utilize the prompt functions provided by Pathway's `pw.xpacks.llm.prompts` module.

Generation

Answer Synthesizing Agent Utilizes a system prompt to generate the answer along with references from source document snippets. It also incorporates feedback from the Critique Agent to improve the output.

Critique Agent The Critique Agent performs the following functions:

- Evaluates the relevancy between the user query and the generated answer by assigning an accuracy score
- Generates feedback for the answer provided in response to the query
- Uses a predefined score threshold to determine if the current answer is sufficient
- If the score is below the threshold, the system increases the k value (number of documents retrieved), enabling adaptive document retrieval
- After a fixed number of attempts, if the relevancy score remains low, the system responds with a fallback message such as, "I don't have enough information to answer that confidently," and provides the user with the most relevant document snippets

For this purpose, we can use `AnswerRelevancyEvaluator` from `llama_index.core.evaluation`, which uses an LLM to generate both a relevancy score and feedback.

In the next loop of answer synthesis, we will also include the feedback from the previous answer, which may help improve the quality of the response.

This approach allows for adaptive document retrieval based on the complexity of the query and avoids hallucinations by providing fallbacks if the answer is not found.

We plan to integrate this Critique Agent within Pathway's `AdaptiveRAGQuestionAnswerer` class or the `answer with geometric_rag_strategy` from `index` function provided by `pw.xpacks.llm.question_answering` by making necessary modifications to the source code.

0.2.3 LLM Usage

Pathway provides several wrappers for LLMs such as LiteLLM through its LiteLLMChat class. LiteLLM support several providers such as Groq and Google AI Studio which we will utilise for using LLMs through APIs. For local usage , we will utilize pathway supported HFPipelineChat class which allows using models from HuggingFace Model hub. We will look out for suitable local options and load the models locally via techniques like quantisation etc.

0.2.4 Multilingual Text Input

For the Query Preprocessing Agent, we will utilize an LLM capable of handling multiple languages (at least English and Hinglish), which will translate the input into a detailed English query. This ensures that all queries flow through the pipeline in English. For the final response, the same model will be used to translate the answer back into the original language of the query.

We have tested Gemini 1.5 Pro, which performed well for this purpose, but we will continue exploring other models that may offer improved performance.

0.2.5 Voice Input

If a voice input is provided, we will first use an STT (speech-to-text) model to convert the speech into text. This multilingual text will then be passed to the Query Preprocessing Agent, which will handle it as described above.

0.2.6 Admin Panel and Query Escalation System

- An authenticated section will be created for admins, allowing them to add news updates to the original PDF store. This ensures that the vector store remains up to date, enabling the chatbot to respond with the most recent and relevant context.

- **Advanced Query Escalation System (if time permits):** It is our responsibility to ensure that every valid query related to IIT Indore is answered accurately. In cases where the chatbot cannot provide an answer due to missing context in the documents, such queries will be escalated to the admin via a dedicated dashboard within the admin panel. The admin will then be able to respond to these queries manually. This setup requires additional infrastructure, including a cloud database (e.g., PostgreSQL) and an authentication system for users.

0.3 Deployment Ideas

We will deploy our frontend on Vercel.com to ensure easy access to our chatbot.

From a production standpoint, we would like to deploy the Dockerized Pathway real-time data streaming pipeline as a microservice, utilizing the `DocumentStoreServer` provided by `pw.xpacks.llm.servers`. However, if we encounter feasibility issues in maintaining the agentic RAG architecture and data-streaming pipeline as separate services—such as increased complexity, deployment challenges, or integration overhead—we may instead wrap both components into a single Flask app. This Flask app will handle HTTP requests from the frontend of our chatbot. We can then deploy the Flask app using Railway.app or Render.com.

0.4 Timeline

Week	Tasks
Week 1	• Curate a comprehensive knowledge base related to IIT Indore • Set up the initial project structure • Conduct in-depth research on Pathway's API documentation and GitHub repositories to explore architecture and performance optimizations
Weeks 2-3	• Explore advanced RAG architectures and identify strategies for improvement • Develop a containerized Pathway streaming pipeline using Docker • Build a Flask-based agentic RAG implementing multi-agent orchestration from scratch
Week 4	• Design and implement the user interface for the chatbot • Perform further enhancements, testing and debugging of the backend pipeline • Integrate fallback mechanisms for handling errors and failure scenarios
Mid-Term Evaluation	
Week 5	• Add authentication support for admin users • Implement document upload capabilities for dynamic knowledge ingestion • Integrate voice input and multilingual support to enhance accessibility • Optimize the pipeline for improved scalability and robustness
Week 6	• Transition to the deployment phase • Finalize enhancements and conduct comprehensive testing across all components
Week 7	• Deploy the frontend on Vercel • Deploy the Flask-based agentic RAG backend on Render.com • Deploy the Dockerized Pathway streaming pipeline on Render.com or Railway.app • Prepare a detailed final project report, summarizing the architecture, implementation, and resources used