Pathway Problem Statement:

About Pathway

Pathway is shaking the foundations of AI by introducing the world's first post-transformer model that adapts and thinks just like humans.

BDH Architecture paper on Hugging Face: https://huggingface.co/papers/2509.26507

The breakthrough architecture outperforms Transformer and provides the enterprise with full visibility into how the model works. Combining the foundational model with the **fastest data processing engine on the market (the framework at your disposal for this year's inter IIT)**, Pathway enables enterprises to move beyond incremental optimization and toward truly contextualized, experience-driven intelligence. We are trusted by organizations such as NATO, La Poste, and Formula 1 racing teams.

Pathway is led by co-founder & CEO Zuzanna Stamirowska, a complexity scientist who created a team consisting of AI pioneers, including CTO Jan Chorowski who was the first person to apply Attention to speech and worked with Nobel laureate Geoff Hinton at Google Brain, as well as CSO Adrian Kosowski, a leading computer scientist and quantum physicist who obtained his PhD at the age of 20 & co-founded SPOJ – one of the earliest popular competitive programming platforms. The company is backed by leading investors and advisors, including Lukasz Kaiser, co-author of the Transformer ("the T" in ChatGPT) and a key researcher behind OpenAI's reasoning models.

Pathway is headquartered in Palo Alto, California, has offices in Paris and Wrocław, and hires exceptional talent remotely as well.

To stay updated and join Pathway's OS community, star the GitHub repositories below.

- github.com/pathwaycom/llm-app → Pathway's LLM xPack and tooling for RAG/agents
- github.com/pathwaycom/pathway → The core Pathway engine
- github.com/pathwaycom/bdh → The post transformer architecture

The Pathway Framework

The Pathway framework is a Python data processing framework designed for analytics and AI pipelines over data streams. It is the ideal solution for real-time processing use cases such as streaming ETL or Retrieval Augmented Generation (RAG) pipelines for unstructured, changing data.

Key Components of this definition:

1. **Python Framework:** Written in Rust for speed and efficiency, Pathway is usable via Python, making it powerful yet simple to use with just Python know-how.

2. **Data Processing:** The Pathway framework excels at processing large-scale, real-time data and is recognized as the world's fastest data processing engine. As a developer, you can use it for tasks like performing JOINS on incoming data streams (real-time data flow) or updating vector/hybrid indexes in real time. These are just simple examples—its potential goes much further.

3. **AI Pipelines Over Data Streams:** The Pathway framework helps AI systems learn from real-time data streams, enabling applications like sentiment analysis, anomaly detection, and RAG pipelines that automatically adapt to incoming data.

.pathway

Problem Statement:

Hackathon Challenge: The Dynamic RAG Playground

Section 1: Introduction: The Dawn of Live AI

The Problem with Stale Knowledge

In the current landscape of artificial intelligence, many applications powered by Large Language Models (LLMs) are constrained by a fundamental limitation: their knowledge is static. Retrieval-Augmented Generation (RAG) systems, while powerful, often rely on a knowledge base that is a mere snapshot in time. This creates a "knowledge cutoff," where an AI assistant can become instantly obsolete. Imagine a financial chatbot unaware of a market-moving announcement made minutes ago, or a customer service bot providing information based on documentation that was updated yesterday. In a world that operates in real-time, these delays are critical failures.

The Paradigm Shift to "Live AI"

A new paradigm is emerging to address this challenge: "Live AI." This represents a fundamental shift from static, retrospective intelligence to dynamic systems that perceive, learn, and reason in real-time. Live AI applications are perpetually synchronized with the latest version of reality, processing information as it is created, modified, or deleted. This hackathon is an invitation to be at the forefront of this transformation.

Introducing Pathway: The Engine for Live AI

The core technology for this challenge is Pathway, a data processing framework designed specifically for building AI pipelines over live data streams. It allows developers to define complex AI workflows that can process information incrementally, enabling extremely low-latency updates. Its unique architecture unifies batch and streaming data, meaning the same code can be prototyped on static files and then deployed to a live data stream, drastically reducing development complexity. This challenge provides an opportunity to harness this powerful engine to build the next generation of intelligent, real-time thinking applications.

Section 2: The Core Challenge: Build a Real-Time Thinking Application

Formal Problem Statement

Your challenge is to design, build, and deploy a Retrieval-Augmented Generation (RAG) application using the Pathway framework that connects to a dynamic, continuously updating data source. The application's core function must be to provide answers that reflect the absolute latest state of the source data, updating its knowledge base and potential responses in real-time as new information arrives, is modified, or is deleted, without requiring manual restarts or batch re-indexing.

Key Requirement: Demonstrable Dynamism

The central evaluation criterion for this challenge is the "liveness" and dynamism of the application. A successful project is not merely a functional RAG pipeline; it is a *dynamic* RAG pipeline that can prove its ability to think in real-time. Participants must be able to clearly demonstrate that when a piece of information in the source data is altered, the application's response to a relevant query changes almost instantly to reflect that alteration.

Example Application Ideas (to inspire, not restrict)

To stimulate brainstorming and showcase the breadth of possibilities, here are several potential application concepts. Teams are encouraged to pursue these or develop their own unique ideas [1]:

- **Live News Analyst:** A chatbot that ingests articles from a real-time news API. It can answer nuanced questions about breaking news stories, with its understanding evolving as new reports, updates, and corrections are published.
- **Real-Time Stock Analyst:** An agent that connects to a stream of financial news or market data feeds. It could answer queries like, "What is the latest market sentiment regarding Company X's new product launch?" based on articles published seconds ago.
- **Dynamic Documentation Helper:** A question-answering bot for a software project that indexes documentation from a collaborative source like a shared Google Drive or SharePoint folder. As engineers update the documentation, the bot's answers become immediately accurate.
- **Social Media Trend Tracker:** An application that monitors a live stream of posts from a platform like Reddit or another social network. It could summarize the latest public sentiment on a specific topic or identify emerging trends.
- **Live E-commerce Inventory Assistant:** A RAG application that connects to a database of product information and inventory levels. It could answer customer questions about product availability and specifications, with its knowledge reflecting the exact current state of the inventory.

Section 3: Data Source Guidelines and Recommendations

Requirement for a Dynamic Source

A core requirement of this challenge is that all projects must connect to a data source that is genuinely dynamic—one that changes over the course of the hackathon. While using a static dataset is acceptable for initial development, the final submitted project must demonstrate its ability to handle a live, evolving data stream.

Recommended Data Sources: Real-Time News APIs

To ensure all teams have access to a high-quality, continuous stream of text data, the use of a free real-time news API is strongly recommended.[2] These services provide a readily available source of new and updated information, perfect for demonstrating the real-time capabilities of your application.

Table 1: Comparison of Suggested Real-Time News APIs

The following table provides a summary of several news APIs that offer free tiers suitable for this hackathon. Teams should review the terms of service for each and select the one that best fits their project's needs.

API Name	Key Features	Free Tier Limits	Data Format
GNews.io [2]	Global sources, 41 languages, historical data access.	100 requests/day, 10 articles/request, 12-hour delay.	JSON
NewsAPI.org [3, 5]	Live top and breaking headlines, 150,000+ sources.	100 calls/day for development.	JSON
NewsData.io [4, 5]	Live breaking news, 7 years of history, sentiment analysis.	500 calls/month, 10 articles/request.	JSON, Excel
Webz.io Lite [5]	Powerful boolean search, sentiment analysis, 30 days of history.	1,000 calls/month, 10 articles/request.	JSON

Alternative Data Sources

Teams are highly encouraged to be creative and are free to use other dynamic data sources. Pathway's robust connector ecosystem supports a wide variety of options.[6] Some alternative ideas include:

- **Cloud Storage:** Use native connectors for Google Drive, SharePoint, or S3 to monitor a folder for new or modified documents.[8]
- **Streaming Platforms:** Connect to a Kafka topic or an MQTT broker to process events from a message queue.[6]
- **Databases:** Use a connector to a database like PostgreSQL and leverage Change Data Capture (CDC) to stream table updates in real-time.[7]
- **Custom Python Sources:** Any data source accessible via a Python library can be turned into a live Pathway table, offering limitless possibilities.[6]

Section 4: Judging Criteria and Weighting

Overview

Projects will be evaluated by a panel of judges based on a video pitch and a review of the submitted code repository. The criteria are specifically designed and weighted to reward projects that deeply embrace and successfully demonstrate the core concept of "Live AI".[9]

Criterion 1: Real-Time Capability & Dynamism (35%)

- **Description:** This is the most critical criterion. How effectively does the application process and react to new or updated data in real-time? Can the team clearly demonstrate that a change in the data source is immediately reflected in the application's output without manual intervention?
- **What Judges Will Look For:** A convincing live or video demonstration of the end-to-end dynamic behavior. Evidence of low latency between a data event and the knowledge base update. A

robust architecture that can handle a continuous stream of data.

Criterion 2: Technical Implementation & Elegance (30%)

- **Description:** How well is the Pathway framework utilized? Is the code clean, well-structured, efficient, and well-documented? Does the system architecture demonstrate a clear understanding of stream processing principles?
- **What Judges Will Look For:** Correct and idiomatic use of the Pathway API. A clean, readable codebase with a clear separation of concerns. A high-quality README.md file that explains the architecture and provides clear setup instructions.

Criterion 3: Innovation & User Experience (20%)

- **Description:** How creative, unique, and compelling is the application's concept and execution? How intuitive and effective is the user interface? Does the project solve a real-world problem in a novel way? [9]
- **What Judges Will Look For:** A creative use case that goes beyond a generic question-answering bot. A functional and clean user interface (a well-designed command-line interface or REST API is sufficient). Evidence of thoughtful prompt engineering.

Criterion 4: Impact & Feasibility (15%)

- **Description:** What is the potential real-world impact of this application? Is there a clear path from this prototype to a production-ready system? Has the team considered the business, social, or practical value of their solution? [9]
- **What Judges Will Look For:** A clear articulation of the problem being solved. A discussion of potential scalability, commercial applications, or social benefits. A polished presentation that effectively communicates the project's value.

Section 5: Submission Requirements and Deliverables

To be eligible for judging, all teams must submit the following items by the specified deadline.

- **Code Repository:** A link to a public repository on a platform like GitHub or GitLab containing all source code and necessary project files.
- **Comprehensive README.md:** The repository's README.md file is a critical document. It must contain:
  - The project title and a concise description of the application.
  - A clear explanation of the application's architecture.
  - Detailed, step-by-step instructions on how to set up the environment, configure API keys, and run the project. This is essential for judges to verify functionality.[10]
- **Video Demonstration (Maximum 3 minutes):** A short, compelling video that serves as the primary pitch for the project. The video must include:
  - A brief introduction of the team and the problem they chose to solve.
  - A clear demonstration of the application in action.
  - **A dedicated segment that explicitly proves the real-time functionality.** For example, showing a new document being added to a monitored folder and then immediately asking the application a question that it can only answer using the information from that new data source.
- **Submission Deadline:** All materials must be submitted via the official hackathon platform by deadline (would be informed via known platforms). Late submissions will not be accepted.

By combining Pathway's core abstractions i.e. connectors, tables, transformations, and the powerful Rust-based incremental engine) with modern AI elements such as deterministic models, incremental training, and agent orchestration, you will demonstrate what real-time AI looks like in practice.

Deliverables at the end

Codebase
A well-structured, production-ready repository using Pathway as the data engine. It should demonstrate both streaming and batch capabilities where relevant, and may integrate additional frameworks for ML, LLM, or automation components.
Code must be modular, documented, and runnable with clear setup instructions for a clean environment. Include configuration files, environment variables, and dependency management.

Documentation
A concise but complete description of the architecture, design decisions, and workflow.
Include:

- A high-level architecture diagram and brief component explanation
- How Pathway is used for streaming, state handling, and incremental computation
- How your system achieves real-time behavior and integrates external tools or APIs
- Notes on scalability, observability, and extension to other use cases or domains

Example Scenario
Demonstrate your solution in one realistic scenario. This could simulate a continuous stream (financial, operational, or customer data) and show how the system reacts, updates, and outputs decisions or insights.
You may script it through a Jupyter notebook, CLI replay, or lightweight dashboard.

Demonstration Pipeline
A running instance or recorded demo illustrating live updates, reasoning flow, and outputs.
Deployment to cloud is optional, but the demo should clearly show streaming ingestion, transformation, and action.
Short video demonstrations are encouraged.

Testing & Evaluation
Basic unit or component tests validating the correctness of core functions.
Optionally, include load or latency tests that illustrate system responsiveness and reliability.

Team Presentation
A short presentation summarizing your concept, architecture, challenges, and key learnings. Highlight how you leveraged Pathway's streaming engine and why your design matters in a real-world context.

Developer Resources and Technical Requirements

To ensure your project demonstrates strong real-time and production-readiness capabilities, all teams must adhere to the following requirements and leverage the official Pathway developer ecosystem.
Exploit pathway to the max. (Problem statement ends)

.pathway

---

1. Live Data Ingestion with Pathway Connectors

Your system must utilize **Pathway's real-time connectors** to ingest streaming data relevant to your chosen use case.
Pathway provides built-in connectors for **files, databases, message queues, APIs, and web sources**, all operating in streaming mode, ensuring results update in real time as data changes.
If your desired data source is not directly supported, you must extend Pathway's ingestion layer by implementing your own connector using the Custom Python Connector. This allows you to adapt Pathway to new sources and contributes to enhancing the engine's capabilities.

Documentation: https://pathway.com/developers/user-guide/connect/connectors-in-pathway
Create a custom Python connector:
https://pathway.com/developers/user-guide/connect/connectors/custom-python-connectors
Python web scraper example:
https://pathway.com/developers/user-guide/connect/python-web-scraping

Artificial Data Streams with the demo Module (in case you find it difficult to access free streaming data APIs):
https://pathway.com/developers/user-guide/connect/artificial-streams

At least one live or simulated data feed must be integrated.
Examples include:

- Subscribing to live market APIs (e.g., Alpha Vantage, Polygon.io)
- Reading transaction streams from Kafka or sockets
- Simulating live events via Pathway's demo utilities

If live data is unavailable, teams may simulate streaming input by replaying static datasets with realistic time intervals.

---

2. Core Concepts

Before starting development, familiarize yourself with Pathway's foundational ideas and architecture. These concepts will help you understand incremental computation, table semantics, and event-driven design principles that power every Pathway pipeline.

Pathway Core Concepts —
https://pathway.com/developers/user-guide/introduction/concepts/#core-concepts

---

3. Streaming Transformations and Feature Engineering

All data transformations must be performed in streaming mode using Pathway's transformation APIs. Your pipeline should support:

- Incremental joins, filters, and aggregations
- Stateful window computations
- Real-time feature engineering for signals and indicators

Documentation: https://pathway.com/developers/user-guide/data-transformation/table-operations
Temporal Data Windows — https://pathway.com/developers/user-guide/temporal-data/windows-manual

Ensure computations are low-latency and modular, with clear separation between ingestion, transformation, and output modules.

---

4. LLM Integration for Real-Time Insights

To make your system interactive and human-centric, integrate **Pathway's LLM xPack** — enabling smooth orchestration of retrieval, summarization, and reasoning over live data.
You may use it for:

- Live retrieval-augmented generation (RAG)
- Automated report generation
- Explainable insights (e.g., credit decision rationale, fraud summary reports)

Documentation: https://pathway.com/developers/user-guide/llm-xpack/overview
Pathway MCP Server: https://pathway.com/developers/user-guide/llm-xpack/pathway_mcp_server

---

5. Mandatory Learning Resources and Templates

These resources will accelerate development and ensure alignment with Pathway's architecture.

Templates
RAG App Templates (YAML): https://pathway.com/developers/templates/
ETL and ML Time-Series Pipelines (Live Data Framework): https://pathway.com/developers/templates/?tab=live-data-framework

Hands-On Tutorials
Vanilla RAG with OpenAI (Python):
https://pathway.com/bootcamps/rag-and-llms/coursework/module-5-hands-on-development/1-first-rag-pipeline/building-with-open-ai
RAG with Gemini:
https://pathway.com/bootcamps/rag-and-llms/coursework/module-5-hands-on-development/1-first-rag-pipeline/rag-with-gemini-and-other-open-ai-alternatives
Real-Time RAG using LlamaIndex:
https://pathway.com/bootcamps/rag-and-llms/coursework/module-5-hands-on-development/3-realtime-rag-with-llamaindex-langchain-and-pathway/implementation-with-llamaindex
Real-Time RAG using LangChain:
https://pathway.com/bootcamps/rag-and-llms/coursework/module-5-hands-on-development/3-realtime-rag-with-llamaindex-langchain-and-pathway/implementation-with-langchain

Advanced Notebooks
Explore the step-by-step cookbooks demonstrating how to combine Pathway's real-time indexing with LangGraph multi-step agent flows:
https://github.com/pathwaycom/llm-app/blob/main/cookbooks/self-rag-agents/pathway_deploy_langgraph_agents.ipynb

If you are only interested in using Pathway as an always up-to-date document store and want to deploy your agents your own way (via Flask, FastAPI, etc.), then check out this cookbook:
https://github.com/pathwaycom/llm-app/blob/main/cookbooks/self-rag-agents/pathway_langgraph_agentic_rag.ipynb

Evaluation & Benchmarks
Evaluating RAG Applications with RAGAS — https://pathway.com/blog/evaluating-rag

Deployment
Docker Deployment Guide —
https://pathway.com/developers/user-guide/deployment/docker-deployment
Persistence and Fault Tolerance —
https://pathway.com/developers/user-guide/deployment/persistence
Licensing Guide (For unlocking Advanced Features) —
https://pathway.com/developers/templates/licensing-guide

Reference Implementations
Option Greeks Computation with Databento —
https://pathway.com/developers/templates/etl/option-greeks/
Real-Time Multimodal Data Processing (Docling) —
https://pathway.com/blog/multimodal-data-processing
La Poste ETA Microservices Case Study — https://pathway.com/blog/pathway-laposte-microservices/

Pathway Community Spotlights — https://pathway.com/blog/?tag=community, https://pathway.com/blog/ai-agents-finance-due-diligence/, https://pathway.com/blog/live-ai-multi-agentic-rag, https://pathway.com/blog/financial-intelligence-with-event-based-state-machine