***Seminar: Xây dựng hệ thống đề xuất món ăn và phân tích bỏ giỏ hàng trên GrabFood theo vòng đời Data Engineer và 6V của Big Data***

**Cuộc hội thoại:**

**DS**: "Chúng mình muốn cải thiện hệ thống đề xuất món ăn trên GrabFood. Cụ thể, chúng mình cần *phân tích hành vi của người dùng để dự đoán món ăn tiếp theo* mà họ có khả năng đặt. Ngoài ra, chúng mình cũng muốn hiểu lý *do tại sao người dùng từ bỏ giỏ hàng khi chưa hoàn tất đơn hàng*."

Phân tích pipeline theo vòng đời Data Engineer

Yêu cầu 1: Dự đoán món ăn tiếp theo mà người dùng có khả năng đặt

Yêu cầu 2: Phân tích lí do người dùng từ bỏ giỏ hàng

**1. Giới thiệu**

GrabFood là nền tảng đặt món ăn trực tuyến với hàng triệu người dùng hàng ngày. Việc tối ưu hệ thống đề xuất món ăn cá nhân hóa không chỉ giúp tăng doanh thu mà còn nâng cao trải nghiệm khách hàng. Bên cạnh đó, tình trạng người dùng từ bỏ giỏ hàng giữa chừng gây thất thoát lớn về doanh thu.

Trong seminar này, chúng ta sẽ phân tích pipeline Data Engineer toàn diện để thu thập, xử lý, phân tích dữ liệu hành vi người dùng, kết hợp với mô hình dự đoán và phân tích bỏ giỏ hàng. Đồng thời, áp dụng nguyên lý 6V của Big Data và lựa chọn công nghệ phù hợp để xây dựng hệ thống hiệu quả, hiện đại.

**2. Pipeline Data Engineer theo vòng đời dự án**

* 1. **SOURCE SYSTEM**
* **Nguồn dữ liệu**
* **Nguồn phát sinh dữ liệu thô từ hệ thống thực tế:**

Dữ liệu hành vi người dùng trên app/web GrabFood (log đặt món, nhấp chọn món, thời gian tương tác, lịch sử đơn hàng).

 Dữ liệu giỏ hàng: trạng thái giỏ hàng (thêm món, xóa món, thời gian từ lúc thêm đến lúc hủy hoặc hoàn tất).

 Dữ liệu tương tác bổ sung: feedback, hủy đơn, thời gian trên trang.

 Dữ liệu người dùng (tuổi, khu vực, thiết bị, lịch sử đặt món).

* **Tính đa dạng và biến động dữ liệu**   
  Dữ liệu phát sinh từ rất nhiều kênh, không chỉ app/web mà còn có thể từ các đối tác thanh toán, đối tác giao hàng, sự kiện bên ngoài như khuyến mãi, lễ hội, thời tiết... Đây là các yếu tố rất quan trọng ảnh hưởng đến hành vi người dùng và đặc điểm đơn hàng.
* **Khối lượng và tốc độ**   
  Số lượng sự kiện người dùng lớn, liên tục, đặc biệt vào giờ cao điểm hoặc dịp khuyến mãi, tạo áp lực cho hệ thống thu thập và xử lý.
* **Ảnh hưởng của 6V:**
  + **Volume**: Hàng triệu sự kiện mỗi ngày, dữ liệu đa dạng từ nhiều kênh (app, thanh toán, giao hàng, đối tác bên ngoài).
  + **Variety** & **Variability**: Dữ liệu đa dạng về định dạng và nguồn, biến động theo mùa vụ, chương trình khuyến mãi, sự kiện.
* **Bổ sung**:
  + Chú ý **đồng bộ hóa thời gian** giữa các nguồn dữ liệu để tránh sai lệch khi phân tích hành vi.
  + Xác định các **key events** quan trọng trong user journey (ví dụ: mở app, xem món, thêm giỏ, thanh toán).
* **Công nghệ tiêu biểu:**
  + Các hệ thống ứng dụng (Mobile App, Web App), database truyền thống, event logging system (ví dụ: Google Analytics, Firebase)
* **Phân tích công nghệ trọng tâm:** **Event Logging System (Google Analytics / Firebase)**
* Ghi lại mọi hành vi người dùng trên app/web theo thời gian thực, hỗ trợ định dạng đa dạng (sự kiện, page views, tương tác)
* Tích hợp dễ dàng, khả năng mở rộng theo người dùng và tương tác
* Đảm bảo dữ liệu được chuẩn hóa ngay từ đầu, tạo nền tảng cho downstream ingestion

***Ghi chú thuyết trình:***

***"Event Logging System là xương sống của tầng source. Nếu không có lớp này, hệ thống không thể hiểu người dùng đang làm gì. Việc chuẩn hoá event sớm giúp downstream tiết kiệm rất nhiều công sức."***

* 1. **DATA INGESTION( Thu thập dữ liệu)**
* Dữ liệu cần thu thập

o **Hành vi người dùng**: click, xem món, thêm vào giỏ, mua hàng, thời gian tương  tác, từ khóa tìm kiếm, thời gian sử dụng ứng dụng, thời gian xem sản phẩm,..

o **Thông tin đơn hàng**: món đã đặt, thời gian đặt, vị trí người dùng, tổng giá trị đơn.

o **Các hoạt động đối với giỏ hàng**: Thêm, xóa và thay đổi số lượng các món ăn  hoặc các đơn hàng được thêm món nhưng không hoàn tất.

o **Thông tin giỏ hàng bị bỏ**: thời gian rời đi, bước rời khỏi, món trong giỏ.

o **Dữ liệu bổ sung**: thời tiết, sự kiện đặc biệt (VD: lễ, bóng đá), chương trình  khuyến mãi.

o **Thông tin thanh toán**: Phương thức thanh toán, việc sử dụng các ưu đãi. o o **Feedback**: đánh giá món, khảo sát sau khi hủy đơn.

* Kết hợp real-time và batch ingestion

Đây là điểm then chốt để xử lý dữ liệu nhanh, phục vụ gợi ý món ăn gần như tức thì, đồng thời vẫn có dữ liệu tổng hợp dùng cho phân tích sâu.

* **Xử lý đa dạng định dạng và nguồn (Variety):**  
  Ngoài log sự kiện, còn có dữ liệu dạng JSON, CSV từ các nguồn bên ngoài (thời tiết, bản đồ), nên cần thiết kế pipeline linh hoạt để thích ứng.
* Công cụ

o **Streaming ingestion**: dùng Apache Kafka, Kinesis hoặc Pub/Sub để thu thập sự kiện theo thời gian thực.

o **Batch ingestion**: thu thập logs hoặc đơn hàng mỗi giờ/ngày qua job ETL. o Tích hợp từ các **nguồn bên ngoài** như API thời tiết hoặc Google Maps để người  dùng có thể theo dõi hành trình đơn hàng theo thời gian thực hoặc thanh toán (các  ứng dụng như Momo, Zalopay, Smartbanking).

* **Ảnh hưởng của 6V**
  + **Velocity**: Kết hợp streaming ingestion để xử lý real-time (Kafka, Kinesis) và batch ingestion (ETL định kỳ) cho dữ liệu tổng hợp.
  + **Variety**: Phải xử lý dữ liệu đa định dạng (log, JSON, CSV, API từ bên thứ ba).
  + **Volume**: Lượng dữ liệu khổng lồ cần pipeline có khả năng mở rộng và phân tán.
* **Bổ sung**:
  + Thiết kế ingestion pipeline **linh hoạt**, hỗ trợ mở rộng dễ dàng theo tốc độ tăng trưởng người dùng.
  + Kiểm soát và **giám sát chất lượng dữ liệu ngay tại bước thu thập** (data quality checks) để phát hiện sớm lỗi.
* **Công nghệ tiêu biểu:**
  + Apache Kafka (streaming), AWS Kinesis, Google Pub/Sub, Apache NiFi (điều phối dữ liệu), ETL tools (Airflow, Talend)
* **Phân tích công nghệ trọng tâm: Apache Kafka**
  + Nền tảng streaming data phân tán, hỗ trợ thu thập dữ liệu real-time, độ trễ thấp
  + Cho phép xây dựng pipeline ingest dữ liệu theo mô hình publish-subscribe, dễ mở rộng theo khối lượng dữ liệu lớn
  + Hỗ trợ giữ lại dữ liệu theo thời gian để xử lý lại khi cần
  + **Liên quan 6V:** Giải quyết tốt **Velocity**, **Volume**, **Variety** do khả năng xử lý luồng sự kiện lớn, đa dạng
  + Có thể kết hợp với hệ thống batch ETL để cân bằng giữa real-time và phân tích sâu
  1. **DATA TRANSFORMATION**
* Tiền xử lý và biến đổi dữ liệu

- Xử lý dữ liệu thô thành dữ liệu hữu ích

- Làm sạch dữ liệu: loại bỏ null, format lại timestamp

- Tính toán các feature như:

o Tần suất đặt món

o Thời gian người dùng rời khỏi giỏ hàng

o Tổng tiền trung bình trong giỏ hàng

- Chuẩn hóa tên các món ăn để việc phân loại và gợi ý món ăn được dễ dàng hơn ( ví dụ “cơm tấm” hay “cơm sườn” được xem như cùng một món)

- Tổng hợp các hành vi người dùng theo từng phiên hoạt động của họ (từ khi mở ứng dụng  đến lúc giao hàng).

- Tính toán các thông số về việc bỏ giỏ hàng hoặc thời gian trung bình hoàn thành đơn  hàng để đo lường độ hiệu quả của hệ thống gợi ý và cải thiện đề xuất.

* **Chú trọng Feature Engineering đa chiều:**  
  Không chỉ tính toán tần suất hay thời gian bỏ giỏ mà còn các feature phức tạp hơn như: xu hướng món ăn theo mùa, mối liên hệ giữa các món ăn được đặt chung (market basket analysis), hoặc phân tích ngữ nghĩa từ khóa tìm kiếm bằng NLP.
* **Xử lý dữ liệu bất thường (Veracity):**  
  Phát hiện và xử lý các giá trị sai lệch (ví dụ GPS bị lệch, phản hồi không hợp lệ) để tránh gây nhiễu mô hình.
* **Phiên người dùng (Sessionization):**  
  Tổng hợp toàn bộ hành vi trong một phiên giúp hiểu rõ ngữ cảnh đặt món hoặc bỏ giỏ.
* **6V áp dụng:**
  + Veracity: Tập trung làm sạch dữ liệu, chuẩn hóa timestamp, loại bỏ outlier.
  + Variability: Tính toán feature cần thích ứng với hành vi người dùng thay đổi theo thời gian.
  + Volume: Xử lý dữ liệu khối lượng lớn để tổng hợp thành các bảng feature cho mô hình.
* **Bổ sung:**
  + Triển khai pipeline tái sử dụng được qua công cụ như dbt để đảm bảo hiệu quả, dễ bảo trì.
  + Tính toán các KPIs cho quá trình từ bỏ giỏ hàng (ví dụ: tỉ lệ bỏ giỏ theo từng bước, thời gian chờ).
  + Chuẩn hóa tên món ăn, gom nhóm món tương tự để tránh phân tán dữ liệu (vấn đề dữ liệu dirty data rất phổ biến với tên món).
* **Công nghệ tiêu biểu:**
  + Apache Spark, Apache Flink, dbt (data build tool), Apache Beam, Airflow (workflow orchestration)
* **Phân tích công nghệ trọng tâm: Apache Spark**
  + Framework xử lý dữ liệu phân tán tốc độ cao, hỗ trợ batch và streaming
  + Hỗ trợ lập trình đa ngôn ngữ (Scala, Python, Java), dễ tích hợp trong hệ sinh thái big data
  + Có khả năng xử lý data cleansing, feature engineering với hiệu năng tốt trên dữ liệu lớn
  + **Liên quan 6V:** Đáp ứng tốt **Volume** và **Veracity** với khả năng xử lý dữ liệu sạch, chuẩn hóa, loại bỏ nhiễu
  1. **DATA SERVING**
* **Phục vụ dữ liệu và mô hình**

- Làm cho dữ liệu được sử dụng cho hệ thống downstream (phía sử dụng) - **Analytics:**

o Phân nhóm người dùng theo hành vi (clustering: người thường đặt 1 món, người  hay xem nhưng không mua...).

o Phân tích dữ liệu theo từng phiên hoạt động của người dùng.

o Trực quan hoá dữ liệu theo dạng phễu để tìm các điểm gây cản trở trong quá trình  hoàn tất đơn hàng.

o Phân tích thời gian để xác định khi nào người dùng hay bỏ giỏ hàng nhất (giai  đoạn chọn món, thanh toán hay vận chuyển).

o Xây dựng tính năng đầu vào cho hệ thống gợi ý:

▪ Món ăn phổ biến

▪ Món ăn gần đây nhất

▪ Giờ ăn quen thuộc

▪ Nhà hàng thường đặt

- **Mô hình máy học:**

o Dùng Collaborative filtering models để gợi ý các món ăn theo từng thời điểm  trong ngày hoặc các món ăn được đặt cùng nhau, dự đoán thời gian người dùng  đặt hàng lại và hiển thị thông báo ứng dụng phù hợp.

o Sử dụng Clustering algorithms để gom nhóm những người dùng có hành vi sử dụng giống nhau.

o Áp dụng các Classification models để dự đoán các khả năng người dùng bỏ giỏ hàng.

o Sử dụng xử lý ngôn ngữ tự nhiên (NLP) để phân tích các từ khoá mà người dùng  tìm kiếm để gợi ý món ăn phù hợp.

- **Reverse ETL**

o Liên tục cập nhật lịch sử đặt hàng để phân tích và gợi ý món cho những lần đặt  hàng kế tiếp.

o Đồng bộ hoá dữ liệu người dùng theo từng nhóm để cải thiện hệ thống gợi ý món. o Cung cấp các gợi ý cá nhân hoá thông qua thông báo ứng dụng hoặc kênh thông  tin như email cá nhân.

o Tự động đồng bộ dữ liệu kết quả phân tích và dự đoán về các hệ thống giao diện người dùng, kênh marketing như push notification, email để tăng hiệu quả tác động.

* **Phân tích nguyên nhân bỏ giỏ hàng:**Sử dụng mô hình phân loại, kết hợp với trực quan hoá funnel conversion để tìm điểm “rớt” nhiều nhất trong quy trình.
* **Theo dõi & giám sát (Visibility):**  
  Dashboard real-time theo dõi hiệu quả đề xuất và tỷ lệ bỏ giỏ hàng giúp phản ứng nhanh với sự biến động.
* **6V áp dụng**:
  + **Value**: Tạo ra các mô hình gợi ý món ăn cá nhân hóa, dự đoán bỏ giỏ hàng chính xác.
  + **Velocity**: Phục vụ dữ liệu và dự đoán gần real-time để tăng trải nghiệm người dùng.
  + **Visibility**: Dashboard giám sát hiệu quả đề xuất và funnel bỏ giỏ hàng.
* **Bổ sung**:
  + Reverse ETL liên tục cập nhật dữ liệu lịch sử lên hệ thống gợi ý và marketing.
  + Ứng dụng các mô hình đa dạng: Collaborative filtering, clustering, classification, NLP phân tích từ khóa tìm kiếm.
  + Định kỳ đánh giá hiệu quả mô hình, rà soát dữ liệu đầu vào để tránh drift.
* **Công nghệ tiêu biểu:**
  + TensorFlow, PyTorch (mô hình ML/AI), MLflow (quản lý lifecycle mô hình), Kubernetes + Docker (triển khai mô hình), Apache Superset/Tableau (dashboard, BI)
* **Phân tích công nghệ trọng tâm: TensorFlow**
  + Framework mã nguồn mở phổ biến cho xây dựng và huấn luyện các mô hình machine learning/deep learning
  + Hỗ trợ đa dạng mô hình (collaborative filtering, RNN, CNN, NLP) phục vụ đề xuất món ăn và phân tích bỏ giỏ
  + Tích hợp dễ dàng với hệ thống phục vụ real-time, hỗ trợ huấn luyện phân tán, fine-tuning mô hình
  + **Liên quan 6V:** Góp phần tạo ra **Value** thông qua mô hình đề xuất chính xác và dự đoán bỏ giỏ hàng hiệu quả
  1. **DATA STORAGE**
* **Lưu trữ dữ liệu**

**Phân loại dữ liệu:**

* **Dữ liệu bán cấu trúc:** như event logs, clickstream, lịch sử tương tác người dùng, dữ liệu giỏ hàng chưa hoàn tất… được lưu trữ trong **Data Lake** để đảm bảo tính linh hoạt và chi phí thấp. Các nền tảng phổ biến:
  + AWS S3
  + Google Cloud Storage (GCS)
* **Dữ liệu quan hệ:** bao gồm thông tin đơn hàng, người dùng, nhà hàng, thanh toán… được lưu trữ trong **Data Warehouse** nhằm phục vụ truy vấn nhanh, phân tích chuyên sâu và báo cáo chính xác. Công nghệ thường dùng:
  + Google BigQuery
  + Snowflake
  + Amazon Redshift
* **Dữ liệu cache real-time:** các thông tin được truy cập thường xuyên hoặc cần phản hồi nhanh, như lịch sử đặt món gần đây, các đề xuất cá nhân hóa được lưu trong bộ nhớ đệm (cache) nhằm giảm độ trễ, tăng trải nghiệm người dùng. Các công cụ cache:
  + Redis
  + Memcached
* **Tối ưu lưu trữ cho truy vấn nhanh (Velocity) và dung lượng lớn (Volume):**
  + Kết hợp **Data Lake** và **Data Warehouse** giúp cân bằng giữa:
    - **Chi phí lưu trữ:** Data Lake có chi phí thấp, lưu trữ linh hoạt cho dữ liệu thô và bán cấu trúc
    - **Hiệu năng truy vấn:** Data Warehouse tối ưu cho truy vấn quan hệ, báo cáo và phân tích phức tạp
  + Tối ưu lưu trữ, xử lý dữ liệu lớn theo mô hình phân tán, cho phép mở rộng theo nhu cầu gia tăng người dùng và dữ liệu
* **Caching Layer:**
  + Sử dụng **Redis** hoặc **Memcached** để lưu trữ tạm thời các kết quả truy vấn phổ biến và các dữ liệu đề xuất nhanh
  + Cache được cấu hình tự động làm mới hoặc hết hạn (TTL - time to live) để đảm bảo dữ liệu kịp thời và chính xác
  + Việc sử dụng cache giảm thiểu số truy vấn trực tiếp đến Data Warehouse, giảm tải hệ thống và tăng khả năng phục vụ số lượng lớn người dùng đồng thời
* **Partitioning & Indexing:**
  + Áp dụng **partitioning** theo các tiêu chí như:
    - Khu vực địa lý (ví dụ: thành phố, quận huyện)
    - Thời gian (ngày, tháng, quý)
  + Partitioning giúp giảm phạm vi dữ liệu khi truy vấn, tăng tốc độ xử lý và giảm chi phí
  + Thiết lập **indexing** trên các trường thường xuyên sử dụng để truy vấn (ví dụ: user\_id, order\_id, món ăn) nhằm tăng hiệu quả tìm kiếm và lọc dữ liệu
  + Cơ chế partition và indexing cũng hỗ trợ hiệu quả cho các công cụ phân tích thời gian thực và batch
* **6V áp dụng:**
  + Volume & Variety: Kết hợp Data Lake (AWS S3, GCS) lưu trữ dữ liệu bán cấu trúc và Data Warehouse (BigQuery, Snowflake) lưu trữ dữ liệu quan hệ.
  + Velocity: Caching layer (Redis, Memcached) để giảm độ trễ truy vấn gợi ý.
* **Bổ sung:**
  + Sử dụng partitioning và indexing theo khu vực, thời gian để tối ưu truy vấn và xử lý.
  + Lưu ý xây dựng chính sách bảo mật dữ liệu người dùng phù hợp quy định.
  + Cân bằng giữa chi phí lưu trữ, tốc độ truy cập và khả năng mở rộng.
* **Công nghệ tiêu biểu:**
  + Data Lake: AWS S3, Google Cloud Storage
  + Data Warehouse: Snowflake, BigQuery, Amazon Redshift
  + Cache: Redis, Memcached
* **Phân tích công nghệ trọng tâm: Snowflake (Data Warehouse)**
  + Dịch vụ data warehouse đám mây cung cấp khả năng lưu trữ dữ liệu lớn, khả năng xử lý truy vấn mạnh mẽ, tự động mở rộng
  + Hỗ trợ phân vùng, indexing tự động, giúp truy vấn dữ liệu nhanh trên dữ liệu quan hệ
  + Tích hợp dễ dàng với nhiều công cụ phân tích và ETL
  + **Liên quan 6V:** Đáp ứng **Volume**, **Velocity**, **Variety** với khả năng quản lý dữ liệu quan hệ và bán cấu trúc

**3. CÁC YẾU TỔ NỀN TẢNG(UNDERCURRENTS) HỖ TRỢ PIPELINE DATA ENGINEER**

**3.1 Security – Bảo mật dữ liệu**

Vai trò và tầm quan trọng:

* Bảo vệ dữ liệu cá nhân, giao dịch của người dùng khỏi các truy cập trái phép và tấn công mạng.
* Tuân thủ các quy định bảo vệ dữ liệu như GDPR, PCI DSS, và luật bảo mật dữ liệu Việt Nam.
* Xây dựng lòng tin người dùng, tránh rủi ro pháp lý và tài chính do rò rỉ dữ liệu.

Ứng dụng thực tế trong pipeline

* **Tại Source System:** Mã hóa dữ liệu đầu vào (data encryption), xác thực người dùng và thiết bị trước khi thu thập dữ liệu.
* **Trong Data Ingestion:** Sử dụng kết nối an toàn (TLS/SSL) khi truyền tải dữ liệu qua Kafka, Pub/Sub.
* **Lưu trữ dữ liệu:** Mã hóa dữ liệu tại rest (encryption at rest) trên Data Lake, Data Warehouse; phân quyền truy cập nghiêm ngặt qua IAM (Identity and Access Management).
* **Giám sát bảo mật:** Thiết lập cảnh báo bất thường (anomaly detection) trên luồng dữ liệu và truy cập hệ thống.

Thách thức và giải pháp:

* **Thách thức:** Quản lý quyền truy cập phức tạp do nhiều bộ phận và hệ thống tham gia.
* **Giải pháp:** Áp dụng mô hình least privilege (quyền tối thiểu cần thiết), audit logs để truy vết.

**3.2 Data Management – Quản lý dữ liệu**

**Vai trò và tầm quan trọng**

* Đảm bảo dữ liệu đầy đủ, chính xác, nhất quán và dễ truy xuất trong toàn bộ pipeline.
* Quản lý metadata để định nghĩa, ghi chú ý nghĩa, nguồn gốc dữ liệu giúp tăng khả năng tái sử dụng và giảm lỗi.

**Ứng dụng thực tế**

* Data Catalog: Sử dụng các công cụ như Apache Atlas, AWS Glue Data Catalog để quản lý metadata.
* Versioning: Áp dụng quản lý phiên bản dữ liệu, đặc biệt với dữ liệu huấn luyện mô hình ML để audit và tái tạo kết quả.
* Data Governance: Thiết lập quy trình phê duyệt, kiểm soát thay đổi dữ liệu.
* Kiểm soát chất lượng dữ liệu: Sử dụng các tool như Great Expectations để tự động kiểm tra data quality trong pipeline.

**Thách thức và giải pháp**

* Thách thức: Dữ liệu đến từ nhiều nguồn, dễ bị trùng lặp hoặc thiếu nhất quán.
* Giải pháp: Áp dụng quy trình chuẩn hóa, validation data ở các bước ingestion và transformation.

**3.3 DataOps – Vận hành dữ liệu**

**Vai trò và tầm quan trọng**

* Tự động hóa phát triển, kiểm thử, triển khai và giám sát pipeline và mô hình dữ liệu.
* Tăng tốc độ đưa sản phẩm ra thị trường, giảm lỗi do con người.

**Ứng dụng thực tế**

* Thiết lập CI/CD pipelines cho ETL và mô hình ML bằng Jenkins, GitLab CI, hoặc CircleCI.
* Giám sát pipeline tự động với các công cụ như Prometheus, Grafana để cảnh báo lỗi kịp thời.
* Sử dụng containerization (Docker) và orchestration (Kubernetes) để đảm bảo môi trường nhất quán và mở rộng linh hoạt.

**Thách thức và giải pháp**

* Thách thức: Đảm bảo pipeline phức tạp hoạt động ổn định trong môi trường đa dịch vụ.
* Giải pháp: Áp dụng monitoring toàn diện, retry mechanism, tự động rollback khi phát hiện lỗi.

**3.4 Data Architecture – Kiến trúc dữ liệu**

**Vai trò và tầm quan trọng**

* Thiết kế hệ thống dữ liệu linh hoạt, mở rộng, tối ưu chi phí và hiệu năng.
* Hỗ trợ dễ dàng mở rộng theo sự tăng trưởng của người dùng và dữ liệu.

**Ứng dụng thực tế**

* **Xây dựng kiến trúc phân lớp rõ ràng:**
  + Source Layer – thu thập dữ liệu thô
  + Ingestion Layer – xử lý đầu vào realtime và batch
  + Storage Layer – data lake và data warehouse
  + Processing Layer – transformation, feature engineering
  + Serving Layer – cung cấp dữ liệu cho downstream, mô hình ML
* **Phân tách rõ ràng giữa compute và storage (Snowflake, BigQuery) để tối ưu chi phí**
* **Thiết kế partitioning, indexing phù hợp để tăng tốc truy vấn và phân tích.**

**Thách thức và giải pháp**

* Thách thức: Giữ cân bằng giữa hiệu suất, chi phí và độ phức tạp hệ thống.
* Giải pháp: Đánh giá định kỳ kiến trúc, sử dụng kiến trúc microservices, event-driven để tăng tính linh hoạt.

**3.5 Orchestration – Điều phối luồng công việc**

**Vai trò và tầm quan trọng**

* Quản lý luồng công việc ETL, xử lý dữ liệu và mô hình đảm bảo thứ tự chạy, tái thực thi khi lỗi.
* Giúp tự động hóa pipeline, giảm thiểu thao tác thủ công.

**Ứng dụng thực tế**

* Sử dụng Apache Airflow, Prefect hoặc Luigi để thiết kế workflow cho ingestion, transformation, serving.
* Tích hợp giám sát trạng thái, alert khi pipeline gặp lỗi hoặc chậm trễ.
* Hỗ trợ chạy theo lịch, theo trigger sự kiện, và xử lý phụ thuộc phức tạp.

**Thách thức và giải pháp**

* Thách thức: Xử lý luồng công việc phức tạp với nhiều bước và phụ thuộc đa chiều.
* Giải pháp: Thiết kế workflow rõ ràng, sử dụng DAG (directed acyclic graph) để quản lý phụ thuộc.

**3.6 Software Engineering – Kỹ thuật phần mềm**

**Vai trò và tầm quan trọng**

* Xây dựng code, script và mô hình có cấu trúc, dễ bảo trì và mở rộng.
* Áp dụng kiểm thử, quản lý phiên bản, tài liệu rõ ràng để đảm bảo chất lượng.

**Ứng dụng thực tế**

* Áp dụng modularization, viết function, class tái sử dụng trong các pipeline.
* Viết unit test, integration test cho pipeline ETL và mô hình ML (pytest, unittest).
* Quản lý code bằng Git, áp dụng code review, pull request để đảm bảo chất lượng.
* Container hóa ứng dụng bằng Docker để đảm bảo môi trường chạy nhất quán.

**Thách thức và giải pháp**

* Thách thức: Đội nhóm lớn phát triển code gây ra xung đột và khó quản lý.
* Giải pháp: Áp dụng quy trình Agile, DevOps, code review nghiêm ngặt, và CI/CD.