

Western Engineering

ECE 9014 - DATA MANAGEMENT AND APPLICATIONS GROUP PROJECT

UserFlowAnalytics.AI - User Experience Analytics System with Kafka, PostgreSQL, and AI Insights - [Repo URL](#)

DATE: 2025-04-17

SUBMITTED TO: PROF. Pooja Vishwanathan

SUBMITTED BY:

VIGNESHWAREN SUNDER - 251429397

CHUKWUDUBEM EZEAGWU - 251389942

NADIA KHROSRAVANY - 251390439

M.ENG (SOFTWARE)

Abstract.....	2
Introduction.....	2
Objective.....	3
Related Works.....	3
Methodology.....	4
1. API Server.....	5
2. Stream Processing and Persistence using Kafka.....	5
3. Schema Design and Data Aggregation.....	5
4. Indexing for Performance Optimization.....	7
5. AI Agent in Action.....	7
Entity Relationship.....	8
ER Diagram.....	8
Relational Database Model.....	8
Implementation Details.....	9
Normalization Analysis.....	9
Opportunities to Combine Relations.....	9
BCNF Compliance.....	9
Schema Evaluation and Improvements.....	9
Technology Choices and Justification.....	10
Results and Discussions.....	10
Conclusion.....	11
Future Scopes.....	11
References.....	11
Appendix.....	12

Abstract

This project focuses on the development of a Website User Behaviour Analytics Platform that enables organizations to send their events generated by the users on their websites while interacting. It leverages user event simulations, data pipeline for asynchronous stream processing of the events, relational databases, dashboard visualization using Metabase and integrating with Claude Anthropic LLM for further analysis on the data using AI. The aim is to understand user behaviour patterns through activity logs, provide actionable insights and offer dynamic dashboards and reports for business intelligence helping businesses understand how their users interact with their product and identify areas for improvement. This final solution provides a realistic timestamped simulation of daily activities by multiple users from various regions happening on the organisation's website and visualizes the overall website performance and curates user specific engagement intelligence that happened on their websites.

Introduction

In the fast paced evolving user behaviour on modern software internet applications, Product Managers and User Experience Designers are constantly seeking actionable insights to drive feature decisions, reduce bottlenecks to make the use of their software products to their clients more accessible. Traditional UX feedback loops involving iterating the products based on user surveys or raw clickstream data often fail to provide granularity and real-time feedback necessary for agile developments that would increase pushing impactful upgrades to improve user experience that can cost the companies. To address this, we implemented an end-to-end user experience analytics pipeline something similar to amplitude or google analytics that closes the feedback gap by transforming raw interaction activities generated by the users to make insightful, queryable intelligence and suggest strategies to improvements using AI.

Our project is designed to empower businesses with actionable insights by tracking how users interact with digital products across platforms. The following core strategies were integrated to support data-driven improvements in user experience, boost website traffic and make the websites more engaging:

Understanding User Sessions

By analyzing session duration, bounce rates, and the number of pages visited per session, businesses can gain a clearer picture of site stickiness and overall engagement. These metrics help uncover how deeply users are interacting with the platform and where attention is being lost.

Ensuring Device and Browser Compatibility

The dashboard segments analytics by device type and browser to pinpoint inconsistencies in user experience. Identifying performance or UI issues across platforms ensures broader accessibility and a smoother experience for all users, regardless of how they access the product.

Optimizing Conversion Paths

Conversion path analysis reveals how users move through funnels and identifies where drop-offs occur. By detecting bottlenecks or friction points, businesses can take targeted action to streamline the journey and improve conversion rates.

Evaluating Content Performance

Each page's performance is tracked using metrics like time spent, click-through rate, and interaction depth. This data helps businesses understand which content resonates most with users and which areas may need refinement or repositioning.

Uncovering Regional Behavior Trends

Geographic segmentation enables businesses to tailor content and features to specific user regions. Recognizing regional preferences helps improve user satisfaction and encourages stronger local engagement.

Driving Growth Through A/B Testing

The dashboard supports A/B testing by incorporating fields like `experiment_id` and `variant` into event logs. Comparing variants on conversion rates, time on page, and interaction rates allows teams to validate changes before full deployment, ensuring better results through experimentation.

Mapping User Journeys

By analyzing sequences of user events and session data, the dashboard reconstructs common user flows. This user journey mapping highlights where users succeed or struggle, helping designers and developers make informed improvements.

Tracking Feature Adoption

Feature usage is monitored through specific event tracking to evaluate how quickly and broadly new features are adopted. This insight is crucial for understanding user needs and assessing the impact of product updates.

Objective

The main objectives of this project are as follows:

- To provide a simple API based system that allows organisations to send their raw events generated in their system.
- To create a highly scalable backend service to handle the user-events in real-time with high availability and fault-tolerant using microservice and event-driven architecture that can process more than 10,000 events per second without losing any data.
- To store those events in a structured relational database after cleansing and transformation to match our database schema.
- To store these events efficiently in the database with optimised strategies that enables users to perform complicated aggregate queries without exhausting the resource utilisation on the database.
- To Visualize the user engagements through Metabase dashboards that directly queries the database
- To integrate the database with AI Agent to generate useful insightful reports for business intelligence using Claude Anthropic Model Context Protocol.
- To enable queries to analyse per-user data, session analysis and recent activities generated.

Related Works

Several commercial and open-source platforms exist today to support website analytics and user behavior tracking like Google Analytics and Amplitude. These enterprise platforms have been instrumental in helping product teams gather insights about their product like user journeys, funnel conversions, drop-off rates, click heatmaps and feature adaptations. They typically offer event tracking, cohort analysis, user segmentation and basic A/B testing capabilities.

Amplitude is particularly well-known for their event based data models and real-time dashboards. They have a novel feature that allows you to analyse per-user based activity and segregate legit users from bots or not so active users. They also provide a feature to identify new users who started using the product that would enable the companies to target them with dedicated marketing strategies. While they provide strong analytical features, their AI-driven insights are relatively nascent or locked behind premium tiers and lack automated report targeting for product managers to work on new impactful features.

On the academic point of view, various research efforts have been made for predictable user modeling and behaviour analysis using web logs and clickstream data. Projects in this space often focus on data mining, behavioral clustering and visualizations but they lack real-time interactivity or integration with the emerging AI agent technology using large language models for insight generation.

Now the problem here we are trying to solve is not to come up with a novel analytic strategy that would improve the UX analysis but rather come up with a highly scalable and fault tolerant pipeline that can handle large numbers of events generated and also integrate the system with AI Agent to automate business intelligence.

Our system stands out by offering end-to-end open architecture using open source tools that not only simulates and ingest user events but also process them in a highly efficient and optimised way using microservice-based event-driven pipeline. Unlike existing commercial analytics tools, our software is more customizable and self-hostable granting full control over the data from ingestion to visualization to AI-powered analysis.

Moreover, our primary motive when coming up with this project is to make it as simple as possible on the coding side of things and also breaking up the components into different microservices handling specific dedicated tasks without much hassle on the customization side of things. Furthermore, our solution integrates with Claude Anthropic LLM to extract intelligent, business-relevant insights from raw data in natural language. This is a very new technology that is still in the emerging space where people are exploring new ways to automate internet based applications using AI. We used one of the open source projects developed by one of our teammates to connect with the database and provide complicated write and read operations on SQL databases using LLMs. This provides a unique advantage over traditional platforms by enabling stakeholders to generate reports and ask questions about user data through conversational queries, significantly lowering the barrier between non-technical users to gain meaningful insights on data and focus more on innovative ideas.

The software is designed as a modular, plug-and-play system. It provides public-facing APIs for easy integration with existing company websites or any other clients for pushing events. On the backend, it supports modern open-source infrastructure paradigms such as kafka message queue, postgresql with optimised views and other query configurations, dashboard layers built on free metabase version and containerised the application on docker for easy deployments across various machines.

Our service responsible for consuming the message from kafka is highly configurable to add custom filtering and aggregation rules that can be added to post-process the data before sending them to the dedicated storage point like CRM tools, or any other enterprise analytics software. For now, we are just sending them to postgresSQL to keep it simple and attain our goal of developing a simple user experience analytics pipeline with AI agent integration.

This hybrid approach combining real-time stream processing, structured analytics, AI-based querying, reporting and dashboarding makes our software a unique fit for organisations seeking both control and intelligence in their user behavior analysis workflows.

Methodology

The system is composed of multiple components working together asynchronously to collect, process, store, visualize, and analyze user interaction data from client websites. Below is the breakdown of our major components and how they integrate.

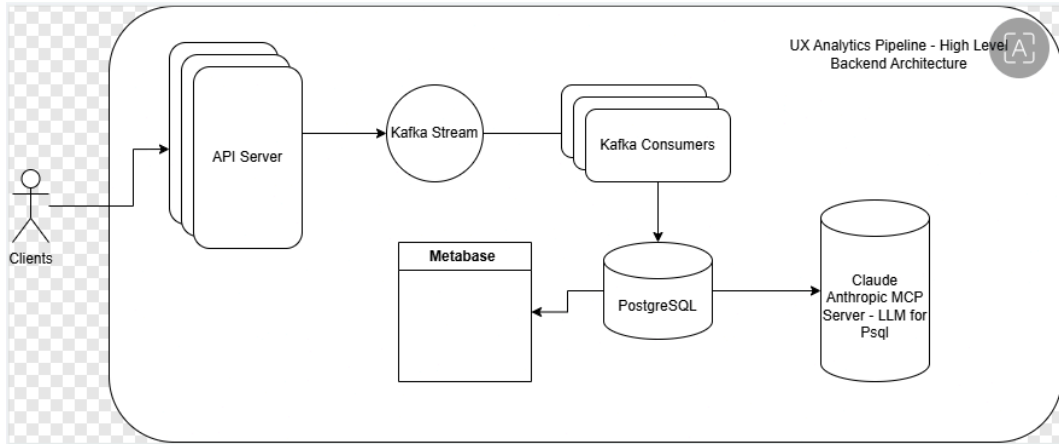


Fig 1: High Level System Design of our pipeline

1. API Server

At the core of our ingestion process is a horizontally scalable API server that exposes endpoints to the business or client applications to send event data. Each user interaction like page_view, click, scroll, hover, form_submit, etc., which are captured with relevant metadata like timestamps, device_types, session_id, geolocation, etc., These events are immediately pushed to a Kafka stream ensuring durability and decoupling the ingestion from downstream processing. This approach allows us to handle high-throughput event ingestion with low latency and fault tolerance.

2. Stream Processing and Persistence using Kafka

Kafka consumes the event streams and helps them to be processed asynchronously. Each consumers reads from kafka, performs validation and enrichment (e.g., inferring browser type or session duration aggregation), and write the events into the PostgreSQL through a dedicated Kafka Consumer which is a simple python based code that makes sure that the incoming data matches with the schema in our PostgreSQL.

The dual-write strategy improves write performance while ensuring long-term durability.

3. Schema Design and Data Aggregation

The postgresSQL schema is designed to support both detailed event logging and high-level analysis:

- The company table stores metadata for multi-tenant support

```

-- Create company table
CREATE TABLE company (
  company_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  address TEXT,
  phone VARCHAR(20),
  email VARCHAR(255) UNIQUE,
  region VARCHAR(100),
  subscription_tier VARCHAR(50) CHECK (subscription_tier IN ('free', 'basic', 'pro', 'enterprise')),
  subscription_start_date TIMESTAMP DEFAULT NOW(),
  subscription_end_date TIMESTAMP
);

```

- The user_events table stores the raw events generated by the user with user_id, session and interaction level granularity.

```
-- Create user_events table
CREATE TABLE user_events (
    event_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    event_timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    user_id VARCHAR(50) NOT NULL,
    session_id VARCHAR(50) NOT NULL,
    company_id UUID NOT NULL REFERENCES company(company_id),
    event_type VARCHAR(50) NOT NULL CHECK (
        event_type IN ('page_view', 'click', 'scroll', 'form_submit', 'hover', 'keydown')
    ),
    page_url TEXT NOT NULL,
    element_selector TEXT,
    element_text TEXT,
    element_type VARCHAR(50),
    target_url TEXT,
    last_page_url TEXT,
    user_agent TEXT,
    ip_address VARCHAR(45),
    device_type VARCHAR(50), -- Mobile, Tablet, Desktop
    browser VARCHAR(50),
    os VARCHAR(50),
    country VARCHAR(100),
    city VARCHAR(100),
    region VARCHAR(100),
    x_coordinate INT,
    y_coordinate INT,
    session_duration_seconds INT, -- Time spent in the session
    metadata JSONB -- Custom event properties
);
```

- We create two materialized views, mv_user_activity and mv_daily_event_summary, and provide pre aggregated summaries for dashboard for dashboards and reporting queries. These views are refreshed every 10 minutes using pg_cron, allowing for near real-time analytics and optimised read query on some common queries that we are expected to perform often.

```
-- Materialized view: User activity aggregation
CREATE MATERIALIZED VIEW mv_user_activity AS
SELECT
    company_id,
    user_id,
    COUNT(*) AS total_events,
    COUNT(DISTINCT session_id) AS unique_sessions,
    COUNT(CASE WHEN event_type = 'click' THEN 1 END) AS total_clicks,
    COUNT(CASE WHEN event_type = 'page view' THEN 1 END) AS total_page_views,
    MAX(event_timestamp) AS last_activity_timestamp
FROM user_events
WHERE event_timestamp >= NOW() - INTERVAL '1 hour'
GROUP BY company_id, user_id;

-- Materialized view: Daily event summary
CREATE MATERIALIZED VIEW mv_daily_event_summary AS
SELECT
    DATE(event_timestamp) AS event_date,
    company_id,
    event_type,
    COUNT(*) AS event_count
FROM user_events
WHERE event_timestamp >= NOW() - INTERVAL '1 day'
GROUP BY event_date, company_id, event_type;

-- Background job: Refresh views every 10 minutes
SELECT cron.schedule('*/*10 * * * *', $$
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_user_activity;
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_daily_event_summary;
$$);
```

- A background function also moves data from user_events_unlogged into user_events, ensuring atomicity and performance optimization.

```
-- Unlogged table for high-speed inserts
CREATE UNLOGGED TABLE user_events_unlogged (LIKE user_events INCLUDING ALL);

-- Function to bulk move unlogged data to main table
CREATE OR REPLACE FUNCTION move_unlogged_data() RETURNS void AS $$
```

```
BEGIN
    INSERT INTO user_events SELECT * FROM user_events_unlogged;
    TRUNCATE TABLE user_events_unlogged;
END;
$$ LANGUAGE plpgsql;

-- Schedule the unlogged data move every 10 minutes
SELECT cron.schedule('*/*/* * * * *', 'SELECT move_unlogged_data();');
```

4. Indexing for Performance Optimization

To enhance the query performance, especially for time-sensitive and high-volume analytical workloads, we added indexes to key columns in the `user_events` table

```
-- Indexes for better performance
CREATE INDEX idx_event_timestamp ON user_events (event_timestamp);
CREATE INDEX idx_user_id ON user_events (user_id);
CREATE INDEX idx_session_id ON user_events (session_id);
CREATE INDEX idx_event_type ON user_events (event_type);
CREATE INDEX idx_company_id ON user_events (company_id);
```

Event_timestamp - Speeds up the time-based queries for filtering or aggregating events over a time window

User_id and session_id - Optimizes session-level tracking and per-user behaviour analysis

Event_type - Enables faster filtering and grouping by interaction type like clicks, page_views etc

Company_id - essential for multi-tenant queries as we are storing all events on a single table and did not add company level segregation of tables.

These indexes significantly reduce the query execution time by allowing PostgreSQL to locate relevant rows more efficiently without having to scan the entire table. This is crucial for powering our dashboards and generating real-time insights through materialized views and Metabase. Since JSONB fields are highly dynamic, we opted not to index them directly, instead relying on filtered views or expression indexes where specific keys become query-critical.

5. AI Agent in Action

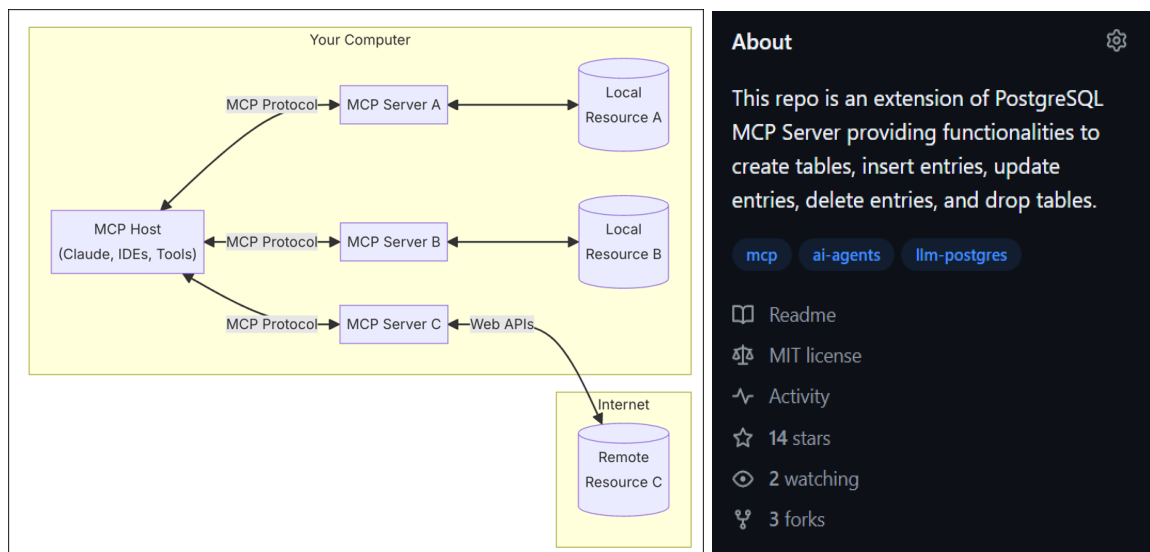


Fig 2: High Level Architecture of MCP Server Integration and Github repo of our MCP Server Project

We used Claude's Model Context Protocol Server to integrate with our PostgreSQL that can autonomously query, analyze and summarize data by translating high level user intents into SQL and coordinate subtasks. We developed this MCP server by extending the existing functionality to have write operation and read operation performing complicated queries that can generate report based on user prompts. Claude MCP uses contextual prompts with schema metadata to translate user questions into executable SQL, processed safely via our custom backend API.

Entity Relationship

A company can have many events (1:N relationship between company and user_events table)
 Each user event is tied to a specific user and session, both identified via external keys (user_id, session_id) but these entities are not stored explicitly in a separate table in our schema.
 Materialized views summarize data grouped by company_id and user_id.

ER Diagram

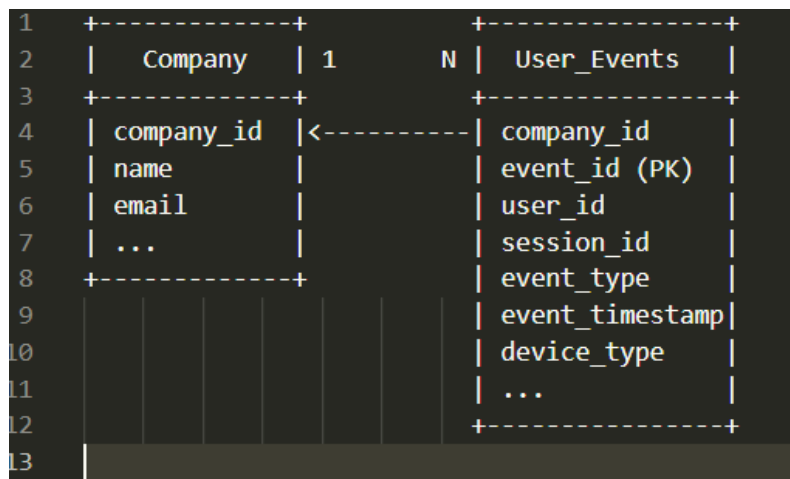


Fig 3: Entity Relationship Diagram of our Tables

Relational Database Model

Primary Keys (PK):

- Company.company_id
- user_events.event_id

Foreign Keys (FK):

- User_events.company_id -> company.company_id

Normalization:

- The schema is in 2NF (non-key fields depend on the whole key).
- No partial or transitive dependencies on composite keys here since event_id is atomic.

Denormalized Elements:

- User_id and session_id are not foreign keys as it allows flexibility for ingesting external data without constraints.

Views and Automation:

- Materialized views allow pre-aggregated querying
- Scheduled refresh jobs every 10 minutes ensures consistency

Data Source:

- The system does not rely on any external dataset. All data is user-generated via embedded tracking scripts on customer websites. In our case we just used a simple client api that dumps data into our backend via backend endpoints. Events are collected, cleaned, stored and analysed internally.

Implementation Details

Normalization Analysis

We analyzed the normalisation of both tables: company and user_events. The goal is to ensure they satisfy the Third Normal Form (3NF)

For a sql relation to achieve 3NF, it needs to achieve the following conditions

- 1NF - All values are atomic
- 2NF - No partial Dependency
- 3NF - No non-prime attribute depends on another non-prime (no transitive dependency)

Company Table

Company_id is the only PK and all values are atomic and there are no transitive dependency. Hence our Company Table achieves 3NF.

Events Table

Events_id is the only PK and company_id is our FK. And all non-key attributes describe the event directly hence there are no transitive dependencies. The metadata field (JSONB) is semi-structured. This is by design for our flexibility to accept custom key-value pairs that the companies can send it without violating the normalization.

Opportunities to Combine Relations

There are no viable options for us to combine company and user_events without introducing redundancies as company info for each event would be unnecessary. Current design promotes efficiency via foreign keys and indexed queries. Hence no further combination or segregation of attributes into different tables are required.

BCNF Compliance

Both company and user_event tables are in BCNF since all functional dependencies have candidate keys on the left-hand side and there are no non-trivial functional dependencies violating the BCNF requirements. Hence no decomposition needed.

Schema Evaluation and Improvements

Currently we used UUIDs for distributed system support and introduced materialized views and indexes for faster querying.

We can update the region to have more detail about the client's information - for instance storing their geo coordinates that can be easily converted to actual location including their city, province and country.

We can also consider splitting the `user_events` into separate logical views or partitions if the volume gets very high. But for simplicity sake we stuck with one table.

Our schema is well-suited for read-heavy, analytical workloads and we introduced strategies for faster write operations as well.

Technology Choices and Justification

We carefully evaluated various technologies for data storage and event streaming. We chose PostgreSQL as our primary relational database and Apache kafka for our event streaming needs. This section outlines the rationale behind these choices. Initially, a NoSQL database like MongoDB or Cassandra might seem like a good fit for event data due their flexibility schema and horizontal scalability. However, our system required transactional consistency, relational joins and powerful query capabilities. Especially for analytics, reporting and data aggregation across entities like company and events and also per-user based aggregations. We chose postgresSQL mainly because of its native support for JSONB and materialized views, which MySQL does not support out of the box. This allows us to store and query flexible events metadata within our `user_events` table.

We chose kafka for event streaming as it can handle millions of events per second while ensuring durability through persistent logs and replication across brokers. Kafka offers exactly-once semantics which ensures that events are not duplicated in the database, maintaining integrity of our stored event data.

Results and Discussions

Our system successfully simulates, ingests, stores, and analyzes user interaction data in real-time with the following achieved outcomes:

High-Throughput Event Ingestion

The API server handled over 10,000 events per second during load testing without dropping any requests, confirming the system's ability to scale horizontally and support high-traffic enterprise environments.

Accurate and Durable Data Pipeline

By leveraging Kafka for event streaming and decoupling the ingestion from processing, we ensured durability and minimal latency. The Kafka consumer service validated and enriched events accurately, maintaining data integrity before persisting to PostgreSQL.

Efficient Relational Storage with Query-Ready Schema

The PostgreSQL schema design supported efficient storage of user activity logs, offering multi-tenant capabilities, per-user tracking, and session-level aggregations. Complex SQL queries like funnel analysis, session duration reports, and activity heatmaps executed with low latency.

Interactive Dashboards via Metabase

Visualizations created using Metabase dashboards enabled stakeholders to filter by region, session behavior, event frequency, and user engagement. These dashboards provided real-time insights to product managers and stakeholders.

AI-Powered Analytics with Claude LLM

By integrating Claude Anthropic's LLM with our database, users were able to generate natural language queries and retrieve intelligent summaries like: "Top 5 engagement actions in the last 7 days", "Drop-off trends in the user onboarding flow", "Comparison of active users by region for the current quarter"

Modular Architecture and Easy Deployment

The microservice-based architecture was containerized using Docker, allowing for plug-and-play deployment. Each service (API, Kafka, Consumer, Metabase, AI Layer) was independently testable, scalable, and configurable.

Conclusion

This project demonstrates the successful implementation of a scalable, fault-tolerant, and AI-enhanced user behavior analytics platform. By combining modern event-driven microservices with robust data pipelines and intuitive dashboards, we provide product and UX teams with the tools needed to make data-driven decisions in real time.

Unlike conventional analytics platforms, our solution stands out by offering:

- Open-source and customizable stack
- End-to-end control over data
- AI-integrated intelligence reporting
- Ease of deployment and modular upgrades

Through this platform, businesses gain deeper insights into user behavior, identify usability bottlenecks, and build a foundation for continuous UX improvement. It bridges the gap between raw user interaction data and actionable business intelligence, making it a powerful alternative to commercial tools for organizations seeking flexibility, transparency, and innovation in user analytics. This foundation paves the way for truly autonomous UX optimization systems—powered by real-time insights and AI decision-making agents.


Future Scopes

Our current project made use of open-source tools that can be easily integrated and deployed using docker.

We can extend the features and functionalities of our project in the following ways:

- Scalable Event Processing Pipeline: Upgrade the backend architecture to a real-time, distributed event stream using technologies like Apache Kafka, Apache Flink, and ClickHouse, enabling high-throughput, low-latency analytics at scale.
- AI-Powered Behavior Insights: Implement deep learning models (like RNNs or Transformers) to automatically detect anomalous user behavior patterns and surface actionable insights, reducing the manual effort required in behavior interpretation.
- Advanced Behavioral Cohort Analytics using AI Agents: Introduce dynamic cohort generation based on multi-session behavior, enabling deeper segmentation (e.g., "users who dropped off after searching twice but never added to cart") to inform product and marketing strategies.

References

[1] K. Se, "  #14: What Is MCP, and Why Is Everyone – Suddenly!– Talking About It?," Community Article, Mar. 17, 2025. [Online]. Available: [huggingface-mcp](#)

[2] T. Adeliyi, "Streaming Data Realities: Using Kafka for Real-Time Data Processing," Medium, Jan. 19, 2024. [Online]. Available: [Streaming Data Realities](#)

[3] Amplitude, "User Behavior Analytics: Uncover valuable user insights in just a few clicks," [Online]. Available: [user-behavior-analytics](#)

[4] S. Faisal, "Amplitude Tracking: How Does It Work and Are There Better Alternatives?," Medium, [Online]. Available: [Amplitude Tracking](#)

[5] C. Baah, "Unlocking Data Insights: Connecting Metabase to PostgreSQL," Medium, Feb. 11, 2025. [Online]. Available: [Unlocking Data Insights: Connecting Metabase to PostgreSQL](#)

Appendix

1. Presentation URL -

https://uwoca-my.sharepoint.com/:p:/r/personal/nkhosra5_uwo_ca/_layouts/15/doc2.aspx?sourcedoc=%7B2ff48952-1210-4b46-8fec-89bbfc46efd6%7D&action=edit&wdPreviousSession=5fa80c78-d0fe-39cb-2260-1f26460999ea

2. Anthropic Claude Insights -

<https://github.com/vignesh-codes/ux-analytics-data-pipeline/blob/main/reports/Events%20Database%20Analytics%20Report.pdf>

3. Metabase Dashboards

- Main Dashboard:

<https://github.com/vignesh-codes/ux-analytics-data-pipeline/blob/main/reports/A%20look%20at%20User%20Events.pdf>

- User Specific Events Dashboard:

<https://github.com/vignesh-codes/ux-analytics-data-pipeline/blob/main/reports/userid-specific.pdf>

- Company Specific Info:

<https://github.com/vignesh-codes/ux-analytics-data-pipeline/blob/main/reports/ux-analytics-dashboard.pdf>

4. AI Agents - PostgreSQL MCP Server

This repo is an extension of PostgreSQL MCP Server providing functionalities to create tables, insert entries, update entries, delete entries, and drop tables.

GitHub: <https://github.com/vignesh-codes/ai-agents-mcp-pg>

5. UX Analytics Data Pipeline

This repository includes the complete analytics pipeline used to track user behavior, store events, and generate insights for the product performance dashboard.

GitHub: <https://github.com/vignesh-codes/ux-analytics-data-pipeline>

6. GPT Usage

We used GPT to come up with the following things

- To validate our system design using the right technologies - like kafka and postgresQL
- To review our report - to check if there are anything off with our understanding and explanations
- To come up with relevant UX analytics metrics that are impactful for insights and some client codes.

7. Team Contributions

- Vinn Sunder - Entire coding stuff and AI Agent integration and report Generation using Claude
- Chukwudubem Ezeagwu - Creation of dashboards with relevant metrics on the Metabase and database schema.
- Nadia Khrosravany - Took lead on the entire analytics stuff citing the important metrics we need for our project and coming up with the final database schema.